НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації та управління

До захисту допущено:
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (вл.ім'я, прізвище)
"___"_____2020 р.

Дипломний проєкт
на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформаційні управляючі системи та технології»
спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: «*Інформаційна система аналізу персоналізованих відгуків учасників навчального процесу*»

Виконав: студент *IV* курсу, групи *ІС-61*

Півоваренко Євгеній Павлович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник *доц., к.т.н., доц. Фіногенов Олексій Дмитрович* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Консультант з графічної документації *доц., к.т.н., доц. Тєлишева Тамара Олексіївна* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Рецензент *доц., к.т.н., доц. Мелкумян Катерина Юріївна* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки та інформаційні технології»

Освітньо-професійна програма «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (вл.ім’я, прізвище)
“___”_____2020 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Півоваренку Євгенію Павловичу
(прізвище, ім’я, по батькові)

1. Тема проєкту «*Інформаційна система аналізу персоналізованих відгуків учасників навчального процесу*»
керівник проєкту *Фіногенов Олексій Дмитрович, к.т.н., доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “*7*”*травня* 2020 р. №1081-с
2. Термін подання студентом проєкту “*01*”*червня* 2020 року
3. Вихідні дані до проєкту
Технічне завдання

4. Зміст пояснювальної записки
1. Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі
2. Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних
3. Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв’язання
4. Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів

5. Технологічний розділ: керівництво користувача, методика випробувань програмного продукту

5. Перелік графічного матеріалу

1. Схема структурна варіантів використань

2. Схема структурна послідовності програмного забезпечення

3. Схема бази даних

4. Схема структурна класів програмного забезпечення

5. Схема структурна компонентів програмного забезпечення

6. Рішення з математичного забезпечення

7. Креслення вигляду екранних звітів

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання *«13» квітня 2020 року*

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Вивчення рекомендованої літератури*		
2.	*Аналіз існуючих методів розв'язання задачі*		
3.	*Постановка та формалізація задачі*		
4.	*Розробка інформаційного забезпечення*		
5.	*Алгоритмізація задачі*		
6.	*Обґрунтування використовуваних технічних засобів*		
7.	*Розробка програмного забезпечення*		
8.	*Налагодження програми*		
9.	*Виконання графічних документів*		
10.	*Оформлення пояснювальної записки*		
11.	*Подання ДП на попередній захист*	*15.05.2020*	
12.	*Подання ДП на основний захист*	*01.06.2020*	
13.	*Подання ДП рецензенту*	*02.06.2020*	

Студент Євгеній ПІВОВАРЕНКО

Керівник Олексій ФІНОГЕНОВ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ДП 6122.00.000 ПЗ	Пояснювальна записка	69	
3	А4	ДП 6122.01.000 ТЗ	Технічне завдання	11	
4	А3	ДП 6122.02.000 ССВ	Схема структурна варіантів використань	1	
5	А3	ДП 6122.03.000 ССП	Схема структурна послідовності	1	
			програмного забезпечення		
6	А3	ДП 6122.04.000 СБД	Схема бази даних	1	
7	А3	ДП 6122.05.000 ССК	Схема структурна класів програмного	1	
			забезпечення		
7		ДП 6122.06.000 ССК	Схема структура компонентів		
			програмного забезпечення		
8	А3		Рішення з математичного забезпечення	1	
9	А3	ДП 6122.07.000 КЗ	Креслення вигляду звітних форм	3	

Зм.	Арк.	ПІБ	Підп.	Дата	ДП 6122 00.000.00		
Розробн.		Півоваренко Є.П.			Відомість дипломного проєкту	Літ.	Лист 1 / Листів 1
Керівн.		Фіногенов О.Д.					
Консульт.						КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-61	
Н/контр.		Телишева Т.О.					
В.о.зав.каф.		Павлов О.А.					

Пояснювальна записка
до дипломного проекту

на тему: Інформаційна система аналізу персоналізованих відгуків учасників навчального процесу

Київ – 2020 року

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проекту складається з шести розділів, 9 рисунків, 12 таблиць, 2 додатків, 25 джерел.

Дипломний проект присвячений розробці комплексу задач класифікації текстів відповідно до згаданих у них осіб методами машинного навчання та обробки природньої мови.

Метою створення системи є забезпечення можливості класифікації анонімних відгуків учасників навчального процесу.

У розділі загальних положень описано процес класифікації текстів, побудовано функціональну модель системи, що проектується, наведений порівняльний аналіз системи та її аналогів, а також визначено мету розробки та задачі, які необхідно вирішити для її створення.

У розділі інформаційного забезпечення наведено детальний опис вхідних та вихідних даних, а також масивів інформації, що генеруються системою.

Розділ математичного забезпечення присвячений обґрунтуванню вибраних методів розв'язання задачі, а також алгоритмів машинного навчання, використаних для створення системи.

Розділ програмного забезпечення описує засоби розробки програмного продукту та його структуру. Описано специфікацію функцій та класів.

У технологічному розділі наведено мету та методи проведення тестування програмного забезпечення, а також наведені його результати.

МАШИННЕ НАВЧАННЯ, КЛАСИФІКАЦІЯ ТЕКСТІВ, ОБРОБКА ПРИРОДНЬОЇ МОВИ, СХОЖІСТЬ ДЖАРО-ВІНКЛЕРА, НАЇВНИЙ БАЄСІВСЬКИЙ КЛАСИФІКАТОР, ДЕРЕВО ПРИЙНЯТТЯ РІШЕНЬ.

ABSTRACT

Structure and scope of work. Explanatory note of diploma consists of six sections, 9 pictures, 12 tables, 2 appendixes, 25 sources.

The diploma project is dedicated to the complex text classification tasks according to the people that could be mentioned with the usage of machine learning and natural language processing.

The system creation aims to provide the possibility to classify anonymous students' review of teaching quality.

In the terms section, the text classification process was described, the functional model of the system was built, comparison analysis of the system and its competitors was provided, also the aim of development with tasks to complete were defined that are needed for system creation.

In the information section, the input and output data were described along with the information the system generates.

The mathematical implementation section is dedicated to the reasoning and describing the methods to solve the task and machine learning algorithms that will be used for system implementation.

The software section describes tools that were used for the actual development, and the structure of the system. Functions and classes specifications were described.

The technological section aligns the aim of the system testing, methods of testing, and testing results.

MACHINE LEARNING, TEXTS CLASSIFICATION, NATURAL LANGUAGE PROCESSING, JARO-WINKLER SIMILARITY, NAÏVE BAYESIAN CLASSIFIER, DECITION TREE.

ЗМІСТ

ВСТУП

Надалі називатимемо задачею визначення іменованих сутностей дві задачі – власне визначення іменованих сутностей (англ. Named Entity Recognition) і зв'язування іменованих сутностей (англ. Named Entity Linking).

Введемо наступні позначення:

Іменована сутність – об'єкт, що представляє собою уособлення сутності предмета або явища, необхідний для абстрактного представлення такої сутності в інформаційній системі; самодостатня одиниця інформації; в контексті даної роботи зазвичай – представлення фізичної особи в інформації.

Документ – зв'язна послідовність слів, що формує собою семантично цілісну та завершену думку.

Сьогодні важко уявити навчальний процес без систем контролю та автоматизованого збору даних. Для покращення якості навчального процесу розробляються нові підходи та методи навчання, аналізуються та оновлюються навчальні матеріали, проводиться двостороннє оцінювання – викладачами студента та викладачів студентами.

Важливим аспектом забезпечення якості освіти є процес комунікації між викладачем та студентом, що дозволить їм знайти та виправити помилки в навчальному процесу для більшої ефективності. Одним із способів такої комунікації є створення відгуків з використанням інформаційних систем загального доступу, наприклад, систем контролю навчального процесу Moodle. При використанні таких систем користувачі, що хочуть залишити відгук, можуть зіштовхнутися з наступними проблемами:

- незручність використання системи;
- можлива деанонімізація відгуків учасниками навчального процесу [1];
- неможливість розповсюдити відгук між усіма учасниками навчального процесу, а тільки між автором та отримувачем.

Отже, системи контролю якості освіти на завжди є зручними для двостороннього зв'язку "студент-викладач", а також потенційно можуть бути використані учасниками навчального процесу для виявлення особи, що залишила відгук. Ці причини привели до появи в інформаційно-комунікаційних системах (месенджерах (англ. "messenger" [2])) спільнот користувачів, що бажають залишити анонімні відгуки про учасників навчального процесу. Такі спільноти адмініструються, зазвичай, третіми особами, не зацікавленими в деанонімізації авторів відгуків та навмисному наклепі на отримувачів відгуку. Варто зазначити, що відгуки в таких спільнотах є загальнодоступними, тож учасники навчального процесу (ті, хто входять в спільноту) можуть вільно переглянути будь-який відгук.

Проблемою даного сформованого підходу до створення відгуків є відсутність формального їх зберігання (у навчальних закладах) та використання для аналізу навчального процесу навчально-виховними структурами навчальних закладів. Також варто зазначити, що такі відгуки мають вигляд неструктурованих текстів без чіткої їх структури чи тегів [2], що дозволили б точно класифікувати відгуки відповідно до осіб, що були у них згадані – найчастіше такі особи і є отримувачами відгуків.

Перед нами постає завдання класифікації неструктурованих текстів та видобування іменованих сутностей з тексту. Для вирішення даного завдання ефективним є метод експертної оцінки, тобто аналіз повідомлень людьми (експертами), що вручну будуть класифікувати відгуки (документи) відповідно до отримувачів. Та варто зважати, що загальний час, необхідний на такий аналіз, пропорційний кількості документів, що повинні бути аналізовані. Враховуючи зростаючу кількість інформації, створення та підтримка систем експертного оцінювання є недоцільною через великі витрати часу, а також, потенційно, фінансів. Вирішити проблему масштабування аналізу відгуків може автоматизація процесу обробки даних відгуків. Дана задачі може бути розглянута як задача класифікації документів.

Задача класифікації документів є однією з пріоритетних задач машинного навчання. Оскільки масиви текстової інформації з кожним роком збільшуються, все актуальніше постає задача автоматичної класифікації документів, а особливо – неструктурованих текстів, джерелами яких є засоби масової інформації, соціальні мережі, тематичні сайти тощо. Збільшення потужності інформаційних систем, а також велика потреба в засобах аналізу текстових даних призвели за останні десятиліття до створення багатьох підходів для обробки та аналізу текстів. Складністю створення таких систем є те, що класифікація текстів – складна задача, в якій досягти хорошої точності на даний момент можна, в основному, створюючи систему під конкретні проблеми, що потрібно вирішити. Це відобразилося у тому, що дані системи, зазвичай, є комерційними та бізнес-орієнтованими і не є доступними для вільного доступу. Також підтвердженням цьому можна назвати "no free lunch" [3] теорему, яка стверджує, що будь-які алгоритми, використані в довільному полі класифікації, поводять себе приблизно однаково, а отже не існує універсального алгоритму для будь-якої задачі машинного навчання і потрібно обирати той, що найкраще підійде для аналізу очікуваних даних.

Дипломний проект присвячений задачі класифікації текстів – відгуків учасників навчального процесу – відповідно до їх отримувачів.

Практичне значення одержаних результатів

Розроблено систему класифікації анонімних неструктурованих відгуків відповідно до отримувача даного відгуку методами машинного навчання та обробки природньої мови.

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

З часом все більше й більше постає питання про покращення якості освіти, покращення підходів до навчального процесу. Одним з аспектів даного питання є питання комунікації учасників навчального процесу – викладачів та студентів. Одним зі способів комунікації, а також можливість формально оцінити якість викладання предмети є відгуки студентів про викладачів, на основі яких можна виправляти помилки в навчальних матеріалах, методах викладання тощо.

Можливість залишати відгуки може бути забезпечена вже існуючими системами контролю навчального процесу, проте з деяких причин їх використання не може бути повноцінним. Саме тому набули поширення спільноти учасників навчального процесу в месенджерах та соціальних мережах. Разом з цим, постає питання визначення отримувача даних відгуків та їх накопичення для можливості подальшого аналізу в навчальних закладах.

Можливим варіантом є створення системи для забезпечення експертного аналізу кожного відгуку, проте варто зважати, що зі збільшенням числа відгуків кількість ресурсів, використаних для аналізу, буде збільшуватись, тож постає питання автоматизації збору та аналізу даних відгуків.

У даній дипломній роботі розроблена система автоматичного збору та аналізу даних відгуків.

Запропоновано метод розв'язання задачі класифікації текстів методами машинного навчання та обробки природньої мови.

1.1.1 Опис процесу діяльності

Для розв'язання задачі класифікації документів відповідно до їх отримувачів, виділяємо наступні етапи в процесі класифікації:

- сформувати масив даних про осіб, необхідних для проведення аналізу;
- зібрати документи для проведення аналізу;
- проаналізувати кожен документ та співставити отримувача документу та сам документ.

Для вирішення задачі використаємо обробку природньої мови для попередньої обробки документів та остаточну класифікацію методами машинного навчання. Є два підходи вирішення задач машинного навчання – з вчителем та без вчителя. Навчання без вчителя може бути використаним для класифікації невідомого класу документів або ж кластеризації документів у класи. Навчання з вчителем припускає, що модель тренується на попередньо розмічених даних, після чого може проводити аналіз даних. Використаємо навчання з вчителем, оскільки нам потрібно співставити конкретних осіб та документи, у яких вони були згадані, але при цьому зважимо на невелику вибірку навчальних даних.

Після розробки системи буде створена система для агрегування та класифікації документів для співставлення учасників навчального процесу та документів.

Система забезпечуватиме наступний процес діяльності для класифікації текстів.

Після розгортання системи, Користувач може сформувати масив інформації про осіб, відповідно до яких потрібно класифікувати повідомлення, параметри агрегації повідомлень зі сторонніх комунікаційних систем.

Також Користувач може здійснити управління збереженими асоційованими тегами, необхідними для збільшення точності класифікації, саме створити (оновити) новий, видалити існуючий або отримати інформацію про дані теги, надалі використовуючи її на свій розсуд.

Структура взаємодії (запитів) Користувача та системи наведена у розділі 2 даної роботи.

Після кожного запиту Користувача сеанс роботи з системою завершується, оскільки кожен такий запит приймається та обробляється незалежно від інших, тож не залежить від запитів, надісланих Користувачем раніше.

Комунікація між Користувачем та системою відбувається за допомогою запитів прикладного програмного інтерфейсу.

1.1.2 Опис функціональної моделі

Для даної системи існує лише один актор – Користувач, що взаємодіє з нею. Варто зазначити, що фізично дані актори передбачаються як автоматизовані системи, котрі взаємодіють з системою, що розроблюється, за допомогою прикладного програмного інтерфейсу.

Опишемо варіанти використання системи Користувачем за допомогою схеми варіантів використання, що представлена на рис. 1.1:

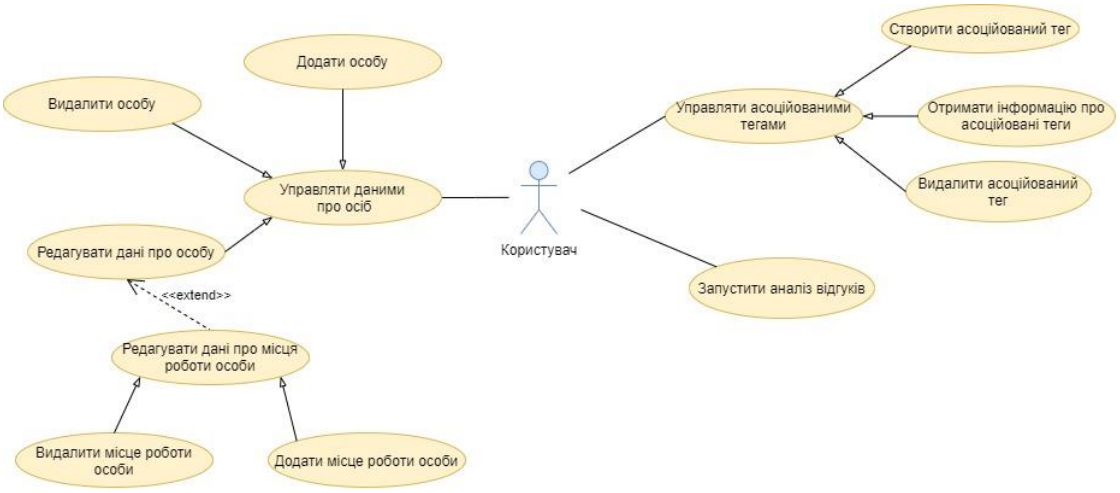


Рисунок 1.1 – Схема варіантів використання системи

Складемо детальний опис варіантів використання, наведений у таблиці 1.1.

Таблиця 1.1 – Варіанти використання системи

Актор	Варіант використання	Опис дії варіанта використання
Користувач	Запустити аналіз відгуків	Користувачу надається можливість надіслати запит на проведення агрегації та аналізу відгуків та отримати відповідь від системи.
	Створити асоційований тег	Користувачу надається можливість надіслати новий тег та асоційовані з ним значення для використання у подальших аналізах системою.
	Видалити асоційований тег	Користувачу надається можливість видалити тег та асоційовані з ним значення з системи.
	Отримати інформацію про асоційовані теги	Користувачу надається можливість отримати масив даних з усіма тегами та їх відповідними асоційованими значеннями.
	Додати особу	Користувачу надається можливість створити нову особу для проведення аналізу.
	Видалити особу	Користувачу надається можливість видалити усі, пов'язані з особою, дані.
	Редагувати дані про особу	Користувачу надається можливість редагувати (змінювати) дані про створену особу

Продовження таблиці 1.1

Актор	Варіант використання	Опис дії варіанта використання
Користувач	Додати місце роботи	Користувачу надається можливість додати місце роботи для створеної особи
	Видалити місце роботи	Користувачу надається можливість видалити місце роботи існуючої особи

Зауважимо, що варіанти використання "Додати особу", "Видалити особу", "Редагувати дані про особу", "Додати місце роботи", "Видалити місце роботи" реалізовані в програмному продукті *не* будуть, оскільки на поточній стадії життєвого циклу системи, необхідно забезпечити відсутність можливості зберігання даних про осіб системою. Дані варіанти використання проектуються для забезпечення можливості додатково зберігати дані про осіб у системі у випадку, якщо під час подальшого використання системи така функціональність буде затребувана, та для мінімізації необхідних подальших змін в структурі системи.

Сформуємо функціональні вимоги для варіантів використання, які наведені в таблиці 1.2.

Таблиця 1.2 – Функціональні вимоги до варіантів використання

Актор	Варіант використання	Функціональна вимога	Пріоритет
Користувач		У випадку недостатності даних для здійснення обробки запиту Користувача, система повинна сповістити про це відповідним повідомленням	Високий
	Запустити аналіз відгуків	Система не повинна зберігати дані про осіб після завершення аналізу відгуків	Високий
		У випадку встановлення отримувачами відгуку декількох осіб з однаковими ПІБ, система повинна не позначити жодного з них як отримувачів відгуку	Високий

1.2 Огляд наявних аналогів

Задача класифікації текстів, як і задача визначення іменованих сутностей, є досить розповсюдженими задачами і для їх вирішення було створено багато систем, як комерційних, так і у відкритому доступі, та інструментів для знаходження іменованих сутностей у тексті на основі контексту повідомлення у вільному доступі немає. Після проведеного дослідження наявності систем надання відгуків був зроблений висновок, що більшість систем, створених для надання відгуків, мають прямий вказівник на об'єкт оцінки і не потребують обробки на предмет визначення сутності, що у них описується, тоді як ми

вирішуємо задачу визначення такої сутності в неструктурованому відгуку (документі).

Складемо таблицю з аналогами програмного забезпечення, що розробляється в даній дипломній роботі, за даними, що надають розробники даного програмного забезпечення [4-6]:

Таблиця 1.3 – Порівняння аналогів системи

Назва	Тип	Засоби для аналізу
spaCy	Програмна бібліотека (Python)	Згорткові нейронні мережі
NLTK	Програмна бібліотека (Python)	Рекурентні нейронні мережі
Stanford CoreNLP	Програмна бібліотека (Java)	Лінійні ланцюги умовних випадкових полів

Варто зазначити, що жоден з даних програмних засобів не розглядає та не може вирішити прямо поставлену задачу визначення осіб (або ж перелічених іменованих сутностей), згаданих у документі. Такий варіант використання можливий, але через безпосередню класифікацію текстів, що дасть поганий результат, оскільки для тренування моделей не існує визначеного набору навчальних даних для кожної особи і такий набір потрібно буде формувати з множини атрибутів, відомих про особу під час кожного окремого запиту.

Також варто зазначити, що для якісної класифікації вищезгаданих бібліотек необхідний великий набір навчальних даних, тоді як поставлена перед нами задача не передбачає попереднє формування розміченого набору даних для навчання.

Дані факти засвідчують унікальність створення програмного засобу цієї дипломної роботи, так як вона може бути в наступному модифікована або ж послужить фундаментом для інших систем.

1.3 Постановка задачі

1.3.1 Призначення розробки

Дана дипломна робота може послужити підґрунтям для створення систем з аналізу та класифікації текстів для визначення об'єктів, згаданих у текстах, на основі контексту документа.

Призначенням розробки є створення та впровадження системи для визначення отримувача анонімного персоналізованого неструктурованого відгуку.

1.3.2 Цілі та задачі розробки

Метою розробки є автоматизація процесу класифікації анонімних відгуків учасників навчального процесу відповідно до отримувача відгуку з використанням засобів машинного навчання та обробки природньої мови.

Для досягнення даної мети, необхідно сформувати та вирішити наступні задачі:

- розробити алгоритм для визначення цільових осіб у неструктурованих документах;
- створити тестовий навчальний набір даних відгуків;
- провести тестування ефективності розробленого алгоритму за допомогою сформованого набору даних;
- визначити контракт прикладного програмного інтерфейсу для взаємодії системи з системами-Користувачами (автоматичного формування та надсилання даних про осіб для аналізу);

- реалізувати можливість для автоматичної агрегації та аналізу відгуків; забезпечити можливість агрегації відгуків з інформаційно-комунікаційних мереж;
- провести тестування розробленої системи на відповідність поставленим вимогам.

Висновок до розділу

У даному розділі був наведений опис предметного середовища, причини, що призвели до створення даної роботи, а також можливі рішення.

Був наведений детальний опис процесів діяльності. Визначені актори, складений опис функціональної моделі, а також визначені функціональні та нефункціональні вимоги, що висуваються до системи, що розробляється.

Був проведений огляд існуючих аналогів та визначені відмінності аналогів та розробленої системи.

Були визначені призначення розробки, а також мету. Для вирішення поставленої мети були сформовані задачі, що мають бути вирішені.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Вхідні дані

Для комунікації Користувача та системи використовується прикладний програмний інтерфейс. Для його опису використаємо мовонезалежний стандарт OpenAPI [7], розроблений для опису прикладних програмних інтерфейсів передачі репрезентативних станів. Складемо специфікацію комунікації Користувача та системи за допомогою даного стандарту у форматі YAML [8]. Зауважимо, що у даній специфікації приведені одразу як вхідні, так і вихідні дані системи. Дана специфікація наведена у Додатку Б.

2.2 Вихідні дані

Так як взаємодія Користувача та системи відбувається за допомогою прикладного програмного протоколу, для її опису використаємо стандарт OpenAPI. Специфікація вихідних даних системи наведена у розділі 2.1 даної роботи.

2.3 Опис структури бази даних

Для уникнення надлишкового (повторного) зберігання даних, а також їх безпеки, система не зберігає інформації про результати аналізу або ж інформацію про осіб, що може використовуватись для аналізу. Виходячи з даних міркувань, складемо схему реляційної бази даних, що представлена на рис. 2.1.

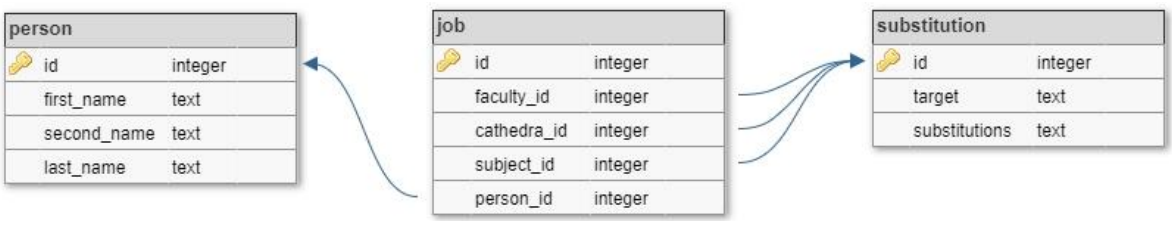


Рисунок 2.1 - Схема бази даних

В структурі бази даних передбачена можливість зберігати дані осіб, відносно яких проводиться аналіз, а також асоційованих тегів. Зауважимо, що, згідно функціональних вимог система не повинна зберігати дані про осіб, та ми все ж проектуємо таблиці реляційної бази для даної інформації у разі необхідності її зберігання, щоб уникнути додаткових міграцій баз даних у майбутньому. Також дана структура дозволяє створити нормалізовану структуру бази даних з мінімально необхідною кількістю інформації.

На схемі представлені наступні таблиці:

- "person" – таблиця з інформацією про особу;
- "job" – таблиця з інформацією про одне з місць роботи особи;
- "substitution" – таблиця з даними про теги та асоційовані з ними значення.

Наведемо детальний опис таблиць, представлений у таблиці 2.1.

Таблиця 2.1 – Опис структури бази даних

Таблиця	Атрибут	Тип	Опис	Приклад
substitutions	id	Ціле число (сурогатний ключ)	Унікальний ідентифікатор запису в таблиці	100500
	target	Рядок символів (унікальний)	Унікальне ім'я, за яким визначений масив асоційованих текстових значень	Факультет інформатики та обчислюваної техніки

Продовження таблиці 2.1

Таблиця	Атрибут	Тип	Опис	Приклад
substitutions	substitutions	Рядок символів	Серіалізований в текстовий рядок масив формату JSON, кожне значення якого можна використовувати як асоціацію з атрибутом "target" під час проведення аналізу	["ФІОТ", "ФИВТ", "ФИОТ"]
person	id	Ціле число (сурогатний ключ)	Унікальний ідентифікатор запису в таблиці	100500
	first_name	Рядок символів	Ім'я особи	Петро
	second_name	Рядок символів	Ім'я по-батькові особи	Петрович
	last_name	Рядок символів	Прізвище особи	Петренко
job	id	Ціле число (сурогатний ключ)	Унікальний ідентифікатор запису в таблиці	100500

Продовження таблиці 2.1

Таблиця	Атрибут	Тип	Опис	Приклад
job	faculty_id	Ціле число (зовнішній ключ)	Посилання на унікальний ідентифікатор тегу (факультет)	100500
	cathedra_id	Ціле число (зовнішній ключ)	Посилання на унікальний ідентифікатор тегу (кафедра)	100500
	subject_id	Ціле число (зовнішній ключ)	Посилання на унікальний ідентифікатор тегу (предмет)	100500
	person_id	Ціле число (зовнішній ключ)	Посилання на унікальний ідентифікатор особи, чия дана робота є	100500

2.4 Структура масивів інформації

Для зберігання необхідної інформації система використовує базу даних, опис структури якої наведений у розділі 2.3 даної роботи. Інші масиви інформації не використовуються.

Висновок до розділу

У даному розділі були визначені вхідні та вихідні дані, які приймає та надсилає система відповідно, їх структуру та тип на усіх рівнях вкладеності.

Визначені структури даних, що будуть опрацьовані системою для усіх варіантів використання.

Була описана структура бази даних, необхідної для функціонування системи, а також дані пояснення з приводу її структури.

Була наведена специфікація сигналів, що приймає та відправляє система, за допомогою стандарту OpenAPI та формату розмітки YAML.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Змістовна постановка задачі

Нехай маємо n категорій іменованих сутностей, що представляють собою унікальних осіб, та m документів, в яких вони можуть згадуватись. Задача полягає у тому, що необхідно визначити, які іменовані сутності згадуються в i-му документі за умови, що кількість сутностей, що може згадуватись в документі:

$$0 \leq K \leq n \tag{3.1}$$

Цю задачу можна розглядати як підзадачу задачі класифікації документів за категоріями, оскільки нам потрібно вказати категорії, до яких документ може належати, якщо ці категорії є категоріями іменованих сутностей.

Тобто, розв'язок задачі можна звести до таблиці виду:

Таблиця 3.1 – Приклад розв'язку задачі в загальному вигляді

Документ	Категорії іменованих сутностей
Документ 1	Категорії 1, 2
Документ 2	Категорія 3
Документ 3	Категорії відсутні
Документ 4	Категорія 2
…	…
Документ m	Категорія n

Розглянемо приклад таких документів:

Таблиця 3.2 – Приклади документів та співставлених осіб

Документ	Категорії іменованих сутностей
Іваненко І.І. – чудовий викладач, його пари – суцільне задоволення.	Іваненко
Іваненко І.І. з факультету #1 – неперевершений викладач. Петренко П.П. чудово проводить практики	Іваненко, Петренко

Для вирішення даної задачі потрібно розбити поставлену задачу на дві підзадачі, а саме:

1. Виділення наявних категорій іменованих сутностей з тексту.
2. Визначення, до якої підкатегорії кожної категорії належить кожна сутність.

Нехай приймемо, що кожна *j*-та категорія представляє собою сукупність *l* підкатегорій, кожна з яких, в свою, чергу, представляє собою особу з певним ім'я, прізвищем, ім'ям по-батькові. Отже, кожна категорія є сукупністю осіб з однаковими ім'ям, прізвищем та ім'ям по-батькові.

Таким чином, розв'язок задачі зводиться до наступного вигляду:

Таблиця 3.3 – Приклад розбиття категорій на підкатегорії у розв'язку

Документ	Категорії іменованих сутностей
Документ 1	Категорія 1, підкатегорія 2; Категорія 2, підкатегорія 3
Документ 2	Категорія 3, підкатегорія 1
Документ 3	Категорії відсутні
Документ 4	Категорія 2, підкатегорія 1
…	…
Документ *m*	Категорія *n*, підкатегорія *l*

Виходячи з описаного, сформулюємо постановку задачі для кожної підзадачі:

1. Визначити, які категорії (множини унікальних іменованих сутностей) згадуються в документі.
2. Визначити, які підкатегорії (унікальні іменовані сутності) кожної категорії згадуються в документі.

3.2 Математична постановка задачі

3.2.1 Математична постановка задачі визначення згаданих у документі категорій

Нехай маємо множину документів:

$$D = \{d_1, d_2, \dots, d_m\}, \tag{3.2}$$

де d_i – документ, $i = \overline{1, m}$.

Нехай наявна множина категорій іменованих сутностей, кожна категорія утворена особами з однаковими прізвищами, іменами та іменами по-батькові:

$$E = \{e_1, e_2, \dots, e_n\}, \tag{3.3}$$

де e_i – категорія іменованих сутностей, $i = \overline{1, n}$.

Задача полягає у визначенні такої функції $G(D, E) \to R_{m \times n}$, де

$$R = \begin{pmatrix} r_{11} & \cdots & r_{1n} \\ \vdots & \ddots & \vdots \\ r_{m1} & \cdots & r_{mn} \end{pmatrix}, \tag{3.4}$$

де $r_{ij} \in \{0,1\}$; 1, якщо i-й документ містить в собі j-ту категорію іменованих сутностей, 0 в іншому випадку.

3.2.2 Математична постановка задачі визначення згаданих у документі підкатегорій

Нехай маємо множину документів:

$$D = \{d_1, d_2, \dots, d_m\}, \tag{3.5}$$

де d_i – документ, $i = \overline{1,m}$.

Нехай маємо категорію іменованих сутностей, що є множиною підмножин, кожна з яких представляє наявні дані про підкатегорію, тобто:

$$C = \{C_1, C_2, \dots, C_n\}, \tag{3.6}$$

де C_i – підкатегорія, представлена множиною векторів параметрів, що зустрічаються в усіх підкатегоріях даної категорії, $i = \overline{1,p}$,

$$C_i = \{c_1, c_2, \dots, c_q\}, \tag{3.7}$$

де c_i – вектор, позначає наявність асоціацій значень будь-якого з категорії з даною підкатегорією та має вигляд:

$$c = \begin{pmatrix} c_1 \\ \vdots \\ c_l \end{pmatrix}, \tag{3.8}$$

де $c_i \in \{0,1\}$, l – кількість унікальних параметрів сумарно по всіх підкатегоріях (тобто вектор представляє усі можливі значення для усіх параметрів для усіх підкатегорій даної категорії); 1 позначає, що для даного

спостереження i-те значення параметрів асоціюється з даною підкатегорією, 0 в іншому випадку.

Задача полягає у визначенні такої функції

$$F(D, C) \to K_{m \times p}, \tag{3.9}$$

де

$$K = \begin{pmatrix} k_{11} & \cdots & k_{1n} \\ \vdots & \ddots & \vdots \\ k_{m1} & \cdots & k_{mn} \end{pmatrix}, \tag{3.10}$$

де $k_{ij} \in \{0,1\}$; 1, якщо i-й документ містить в собі j-ту підкатегорію іменованих сутностей, 0 в іншому випадку.

3.3 Обґрунтування методу розв’язання

3.3.1 Обґрунтування методу розв'язання визначення категорій у документі

Для визначення збірних категорій у документі, що є множиною підкатегорій названих іменованих сутностей, можемо використати засоби обробки природніх мов, що дозволить виділити необхідні дані з документа на основі правил та аналізу даного документа з відносно невеликими обчислювальними витратами.

Оскільки кожна підкатегорія складається з сутностей, що представляють собою осіб з певними іменем, прізвищем, іменем по-батькові, це дозволяє спростити аналіз документа, враховуючи, що документи написані однією природньою мовою і існують правила, за якими повинні бути зазначені дані про особу, щоб їх можна було проаналізувати, максимально наближені до згадування таких даних у неструктурованих документах.

Для проведення виділення іменованих сутностей використаємо порівняння наборів слів за допомогою схожості Джаро-Вінклера (англ. *Jaro-Winkler similarity*).

3.3.1.1 Обґрунтування методу схожості Джаро-Вінклера

Алгоритм схожості Джаро-Вінклера використовується для порівняння двох текстових рядків на схожість, рахуючи кількість перестановок, які необхідно зробити, та кількість символів, які збігаються. Даний алгоритм дає оцінку схожості двох рядків в діапазоні [0,1], а не тільки кількість перестановок або замін, що дозволить встановити значення, що буде достатнім, щоб вважати, що рядки будуть одним і тим самим словом. Це дозволить врахувати неправильне написання даних у неструктурованих документах. Вінклер модифікував оригінальний алгоритм Джаро, ввівши припущення, що схожість рядків на початку є більш суттєвою за схожість наприкінці, що дозволяє ефективно використовувати даний метод для порівняння імен, що ідеально підходить для поставленої перед нами задачі.

Перевагами алгоритму є відносно невелика обчислювальна вартість, що є значним фактором при великій кількості даних для аналізу.

3.3.2 Обґрунтування методу розв'язання визначення підкатегорій у документі

Задача визначення підкатегорій покладається на виявленні тегів у документі, що можуть належати особі з певної категорії. Для цих цілей використовуються засоби обробки неструктурованих документів та пошук слів у документі, що можуть бути тегами для будь-якої особи з категорії. За множиною знайдених тегів потрібно провести аналіз й визначити, яка особа з категорії згадується в документі. Для цього використовуються методи машинного навчання.

3.3.2.1 Обґрунтування методу наївного Баєса

Класифікатор наївного Баєса є простим ймовірнісним класифікатором, побудованим на основі (наївного) припущення про незалежність параметрів для прогнозування.

Такий класифікатор добре підходить до поставленої перед нами задачі класифікації особи з категорії, оскільки може бути використаний для невеликої кількості навчальних даних, що забезпечується його ймовірнісною природою.

Перевагами наївного Баєсівського класифікатора є простота реалізації, невелика обчислювальна вартість, здатність тренуватись на невеликій кількості навчальних даних; він добре працює для багатокласової класифікації.

Недоліками класифікатора є припущення про незалежність параметрів, що, загалом, в прикладних задач зустрічається рідко, яке може привести до гірші, порівняно з іншими типами класифікаторів, результати, при реальній залежності параметрів класифікації.

3.3.2.2 Обґрунтування методу дерева рішень

Дерево рішень (алгоритм ID3 [9]) – класифікатор, що використовує правила прийняття рішень, виведені з спостережень за даними. Атрибути розподіляються за ієрархією, враховуючи частість їх повтору для кожної підкатегорії й будуючи таким чином ієрархію атрибутів, що базується на булевій умові входження атрибуту в конкретний екземпляр даних. Після побудови правил, класифікація для кожної підкатегорії починається з кореневого вузла (правила), після чого продовжує перевіряти правила знизу за гілкою дерева, спускаючись до листків, які і є конкретним екземпляром даних (іменованою сутністю).

Такий класифікатор здатний до хорошої класифікації осіб за параметрами, враховуючи явну ієрархічну впорядкованість даних (факультет,

кафедра), що подаються класифікатору для навчання, та відносно невелику кардинальність [10] множини атрибутів.

Перевагами такого класифікатора є простота реалізації; невелика обчислювальна вартість; представляє собою зрозумілий набір правил для класифікації; добре підходить до задач багатокласової класифікації.

Недоліками класифікатора є можливість перенавчання класифікатора; за незбалансованого навчального набору даних дерева можуть будувати упереджені правила, що призводить до домінування одного класу альтернатив над усіма іншими.

Обрані методи розв'язання задачі компенсують недоліки одне одного, що можуть проявитися виходячи зі специфіки даних, що будуть аналізуватись. Баєсівський класифікатор дасть повинен надати кращий результат у випадку, якщо дані мають невелику залежність, дерево прийняття рішень – навпаки, повинен показати кращий результат у випадку, коли дані мають досить сильну залежність.

3.4 Опис методів розв’язання

3.4.1 Опис методів знаходження категорій у документі

Для знаходження категорій у документі необхідно виділити алгоритм, за яким документ буде аналізуватись. Проводиться ітеративний пошук по усіх словосполученнях, що можуть бути потенційно інформацією про особу. Даний пошук проводиться за допомогою регулярних виразів, виділяючи групи слів, визначені за певними правилами. Дозволяє зробити це природа даних, що ми шукаємо – оскільки ми шукаємо словосполучення, що позначають певну особу, а саме ім'я, прізвище, ім'я по-батькові, то можемо виділити правила запису даної інформації та очікуємо, що в документі дана інформація буде описана в одному з визначених нами форматах.

Далі для словосполучень може проводитись нормалізація слів – стеммінг та лематизація [11]. Нормалізація може бути як на основі правил (наприклад, зміни відмінку слова), так і на основі словників. Після цього відбувається порівняння нормалізованих даних з даними кожної особи, щоб з'ясувати, яка особа згадана у даному словосполученні.

Варто зазначити, що через неструктурованість документів або помилки в написанні даних постає питання неможливості проведення нормалізації за допомогою правил, оскільки дані не будуть підпадати під жодне з них; а через унікальність даних постає питання неможливості застосування словників для нормалізації, оскільки є ймовірність того, що дані не містяться в жодному з словників. На допомогу приходить метод схожості Джаро-Вінклера, що дозволить після проведення (або не проведення) нормалізації порівняти текстові рядки з певним рівнем довіри сказати, що дані, отримані з документу, є такими ж для особи, що ми порівнюємо з отриманими. Після аналізу документу та формування списку згаданих осіб відбувається групування осіб з однаковим іменем, прізвищем, ім'ям по-батькові в категорії та наступний етап алгоритму.

3.4.1.1 Опис методу схожості Джаро-Вінклера

Метод Джаро-Вінклера підраховує схожість між двома текстовими рядками, даючи нормалізовану оцінку схожості між 0 (схожість відсутня) до 1 (ідентичні рядки).

Опис даного методу описаний в роботі [12], тут наведемо опис основних положень алгоритму.

Нехай дано два текстові рядки, $s = a_1 \dots a_K$ і $t = b_1 \dots b_L$. Символ a_i є спільним для s і t, якщо існує символ $b_j = a_i$ такий, що $i - H \leq j \leq i + H$, де

$$H = \frac{\min(|s|, |t|)}{2}, \tag{3.11}$$

де $|s|$ – довжина рядка s, $|t|$ – рядка t.

Нехай $s' = a'_1 \dots a'_K$ - послідовність символів з s, що входить також в t. Нехай $t' = b'_1 \dots b'_L$ - послідовність символів з t, що входить в s, тобто $s' = t'$. Таким чином визначаються підрядки, що є еквівалентними для обох рядків на початку. Приймемо $T_{s't'}$ - половина кількості перестановок між s' і t'. Тоді формула схожості рядків Джаро набуває вигляду:

$$Jaro(s,t) = \frac{1}{3} \cdot \left(\frac{|s'|}{|s|} + \frac{|t'|}{|t|} + \frac{|s'| - T_{s't'}}{|s'|} \right) \qquad (3.12)$$

Вінклер у своїй модифікації пропонує ввести додатковий параметр в обчислення, дозволивши врахувати значущість перших символів рядка (що є значущим фактором для порівняння імен, оскільки перші літери, зазвичай, найбільш важливі для порівняння).

Модифікація детально описана в роботі [13], наведемо основні положення.

Приймемо, що l – довжина найбільшого спільного префіксу текстових рядків s і t. Введемо параметр $p \leq \frac{1}{L}$, де L – найбільша дозволена довжина спільного префіксу ($l \leq L$). Зазвичай, L пропонується обмежувати на основі відомих даних про рядки, що будуть порівнюватися, для отримання найкращого результату. Важливим аспектом є значення p, що залежить від даної довжини. Його обмеження забезпечує те, що оцінка схожості буде ≤ 1. Сам Вінклер у своїй роботі рекомендує використовувати $L = 4$, $p = 0.1$.

Модифікована формула за Вінклером:

$$Jaro - Winkler(s,t) = Jaro(s,t) + l \cdot p \cdot [1 - Jaro(s,t)] \qquad (3.13)$$

Таким, чином у даній формулі буде надаватись більша оцінка рядкам, які мають спільні символи на початку, що є важливою деталлю для порівнювання імен та прізвищ.

3.4.2 Опис методів знаходження підкатегорій у документі

3.4.2.1 Опис методу наївного Баєса

Розглянемо основні положення наївного Баєсового класифікатора, особливості якого розглянуті в роботі [14], адаптувавши постановку задачі та позначення до поставленої перед нами задачі. Даний класифікатор базується на ідеї незалежності спостережень випробувань та значень параметрів, що виникають у даних випробуваннях.

Надалі називатимемо *зразком* об'єкт, що потрібно класифікувати згідно наявних підкатегорій (віднести до однієї з них).

Нехай є документ d та категорія $C = \{C_1, C_2, \dots, C_n\}$, що входить в нього (див. розділ 3.2.2). Ціллю класифікації є знайти, яка підкатегорія $\hat{c}$ входить в даний документ (оскільки підкатегорія може бути одна, це рівносильно визначенню задачі належності документу підкатегорії).

Підходом наївного Баєсового класифікатора є знаходження апостеріорних ймовірностей належності зразка кожній з підкатегорій та обрання тієї з них, для якої ця ймовірність найбільша, тобто:

$$\hat{c} = arg \max_{C_i \in C} P(C_i|d) \tag{3.14}$$

Далі необхідно використати формулу Баєса для знаходження ймовірності події:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{3.15}$$

Таким чином, можемо записати формулу (3.14) наступний чином:

$$\hat{c} = arg \max_{C_i \in C} \frac{P(d|C_i)P(C_i)}{P(d)} \quad (3.16)$$

Оскільки знаменник $P(d)$ є спільним для усіх підкатегорій, він буде зменшувати ймовірності знаменника в константну кількість разів для усіх ймовірностей, проте відношення між апостеріорними ймовірностями при цьому не зміняться. Так як аналіз проводиться для одного й того ж документу, ймовірність його появи між підкатегоріями не зміниться, отже можемо прийняти $P(d) = 1$ (рівносильно відкиданню знаменника через властивості ділення). В такому разі, формула (3.16) прийме вигляд:

$$\hat{c} = arg \max_{C_i \in C} P(d|C_i)P(C_i) \quad (3.17)$$

Дана формула й використовується для класифікації, проте безпосередньо обчислити дані ймовірності важко, тож необхідно ввести спрощення, які дозволять обрахувати ймовірності за допомогою ознак та їх ймовірностей. Оскільки кожен документ складається з певних атрибутів, які ми виділяємо з нього (у класичній задачі класифікації текстів – це слова, в нашому випадку – значення атрибутів, видобуті з документу засобами обробки природньої мови), отже кожен документ можемо представити у вигляді таких атрибутів:

$$d = \{f_1, f_2, \dots, f_n\} \quad (3.18)$$

Формула (3.17) прийме вигляд:

$$\hat{c} = arg \max_{C_i \in C} P(f_1, f_2, \dots, f_n | C_i) P(C_i) \quad (3.19)$$

Для формування правила вирахування ймовірності входження атрибуту в документ, який впливає на вибір підкатегорії, скористаємось наївним Баєсівським припущенням, що кожне значення атрибуту впливає на результат класифікації незалежно. Завдяки такому припущенню незалежності можемо перетворити:

$$P(f_1, f_2, \dots, f_n | C_i) P(C_i) = P(C_i) \prod_{f \in d} P(f | C_i) \quad (3.20)$$

Формула (3.19) прийме вигляд:

$$\hat{c} = arg \max_{C_i \in C} P(C_i) \prod_{f \in d} P(f | C_i) \quad (3.21)$$

Формула (3.21) використовується для класифікації підкатегорій у документах.

Для проведення класифікації необхідно провести навчання класифікатора, щоб отримати ймовірнісні оцінки для $P(C_i)$ та $P(f_i)$.

$$P(C_i) = \frac{N_i}{N}, \quad (3.22)$$

де N_i – кількість навчальних екземплярів, що належать підкатегорії C_i, N – кількість усіх навчальних екземплярів у категорії.

$$P(f_k|C_i) = \frac{W_{ki}}{\sum_{j=1}^{l} W_{ji}}, \quad (3.23)$$

де W_{ki} – кількість потраплянь тегу f_k в навчальні дані підкатегорії C_i, $k \in [1, l]$; W_{ji} – кількість потраплянь тегу f_j в навчальні дані підкатегорії C_i.

Таким чином, підрахувавши кількість навчальних екземплярів підкатегорії та входжень значень окремих атрибутів в категорію, можемо отримати ймовірнісні оцінки, що будуть використані при класифікації зразка.

3.4.2.2 Опис методу дерева прийняття рішень

Розглянемо основні положення дерева прийняття рішень, особливості якого розглянуті в роботі [15], адаптувавши постановку задачі та позначення до поставленої перед нами задачі. Даний класифікатор базується на створенні правил за навчальними даними, за якими проводиться подальша класифікація зразків.

Для визначення правила, яке буде слугувати для подальших кроків класифікації в даному вузлі дерева (що будується), обирається той атрибут, який дає найбільший *інформаційний приріст*, тобто:

$$\hat{g} = \max_{A \in d} G(C, A), \quad (3.24)$$

де A – атрибут (усіх) підкатегорій категорії C.

Кількісна оцінка інформаційного приросту обчислюється за формулою:

$$G(C, A) = E(C) - \sum_{a \in A} \frac{|C_a|}{|C|} E(C_a), \quad (3.25)$$

де $a \in A$ – множина всіх значень атрибуту A; $|\mathrm{C}_a|$ – кількість екземплярів даних підкатегорії C_a в навчальній вибірці; $|\mathrm{C}|$ – кількість екземплярів даних усіх підкатегорій категорії C; $E(C)$ – числова оцінка *ентропії* категорії C.

Числове значення ентропії обчислюється за формулою:

$$E(C) = -\sum_{i=1}^{n} p_i \log_2 p_i, \tag{3.26}$$

де

$$p_i = \frac{|\mathrm{C}_i|}{|\mathrm{C}|}, \tag{3.27}$$

Далі задача побудови дерева зводиться до задачі визначення атрибутів, на яких потрібно проводити розмежування значень – створення вузлів дерева, в яких буде відбуватись подальша класифікація екземплярів за певними атрибутами.

Після побудови дерева, для того, щоб класифікувати зразок, алгоритм починає перевірку правила в кореневому вузлі, послідовно спускаючись по ієрархії дерева, поки не досягне якогось з листків. Даний листок і буде класифікованою підкатегорією.

Висновок до розділу

Під час написання даного розділу були введені необхідні поняття, поставлена змістовна та математична постановка усіх підзадач, на які була декомпозована задача.

В змістовній постановці задачі наведений стислий опис поставленої задачі та підзадач, необхідний для розуміння контексту засобів її подальшого вирішення.

В математичній постановці задачі були сформульовані математичні відображення поставлених задачі та підзадач.

В обґрунтуванні методів розв'язання задачі наведене коротка характеристика та стислий огляд того, як запропоновані методи розв'язання можуть вирішити поставлену перед нами задачу, наведені переваги та недоліки методів розв'язання.

В описі методів розв'язання наведений розширений опис кожного з методів розв'язання задачі, описане їх формальне математичне відображення.

4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Засоби розробки

Для написання даної роботи було використано велику кількість програмних засобів з розробки, розгортання програмного забезпечення, а також інструменти для управління базами даних, текстові редактори, засоби створення діаграм.

Наведемо список та короткий опис описаного програмного забезпечення:

- macOS (Catalina) – операційна система, розроблена компанією Apple, створена для апаратних пристроїв даної компанії, таких як портативні комп'ютери, док-станції, персональні комп'ютери. Входить в систему сімейство операційних систем Apple OS X [16], що розробляються для пристроїв даної компанії. Операційна система має широкі можливості розробки, так як підтримується більшістю сучасних виробників програмного забезпечення. Під час виконання даної роботи дана операційна система була використана як основна;
- PyCharm [17] – середовище розробки компанії JetBrains, що призначене для розробки на Python. Забезпечує статичний аналіз коду, можливості рефакторингу коду одночасно у всьому проекті, вбудований налагоджувач (дебагер), підтримку роботи з базами даних, підтримку управління системами керування версіями файлів, підтримку розгортання системи за допомогою засобу контейнеризації Docker. Під час розробки даної роботи слугував як основний засіб розробки;
- Visual Studio Code [18] – середовище розробки, створене компанією Microsoft як альтернатива іншим середовищам, основною перевагою якого повинна була стати легкість використання, а також мала кількість ресурсів, що споживається. Підтримує встановлення

повноцінних доповнень, що дозволяє персоналізувати середовище без зайвого ускладнення стандартного дистрибутиву. Під час розробки даної роботи слугував як допоміжний інструмент для роботи з файлами даних для аналізу;

- Draw.io [19] – веб-сервіс для роботи з діаграмами. Має широкий спектр наявних позначень та графічних об'єктів для створення діаграм в процесі розробки програмного забезпечення та визначення вимог, в тому числі засоби побудов UML-діаграм. Використаний для створених діаграм;
- Postman [20] – програмне забезпечення для тестування прикладних програмних інтерфейсів. Широко застосовується в розробці програмного забезпечення завдяки простоті та великим функціональним можливостям. Postman дозволяє створювати прототипи програмних контрактів, проводити як окреме тестування індивідуальних запитів, так і створювати користувацькі сценарії використання. Оскільки система в даній розробці використовує прикладний програмний інтерфейс для взаємодії з Користувачем, Postman слугує як основний засіб тестування розробленої системи;
- Git – розподілена система керування версіями файлів, що дозволяє проводити поетапну розробку програмного продукту, використовуючи абстрактні сутності – коміти та гілки, які є конкретними версіями файлової системи проекту в даний момент часу та сукупністю пов'язаних комітів відповідно. Дозволяє розробляти застосування, проводити тестування, виправляти помилки після тестування, а вже потім вносити остаточний код в основну версію програми. В даній роботі є основною і єдиною системою керування версій;
- Github [21] – веб-сервіс, що є сховищем для зберігання Git-репозиторіїв (проектів). Має широкі можливості для спільного та

індивідуального зберігання проектів. В даній роботі використовується як засіб для резервного зберігання програмних файлів продукту;

- PostgreSQL (10) [22] – об'єктно-реляційна база даних, широко використовується для індивідуального та комерційного використання. Має активну підтримку з боку технологічних компаній, так як являється безкоштовною і має широкі функціональні можливості. В даній роботі використовується як основна та єдина реляційна база даних;
- Docker [23] – програмне забезпечення для контейнеризації застосунків, що дозволяє розгортати систему або комплекс систем з однієї точки розгортання. За допомогою даного засобу системи можуть бути представлені у вигляді мікросервісів, кожен з яких розгортається та працює окремо від інших. Для даної роботи Docker – основний і єдиний засіб контейнеризації системи та розбиття її на, власне, систему та інші компоненти, які вона потребує (наприклад, база даних, яка теж представлена окремим контейнером);
- Make [24] – утиліти для автоматичного розгортання програм. Є надбудовою над системною консоллю операційної системи і служить для спрощення часто повторюваних дій при розробці та розгортанні застосувань;
- Python [25] – мова програмування високого рівня, широко використовується навчальних, наукових та комерційних цілях. При усій своїй простоті, дана мова ефективна в плані створення прототипів наукових систем великої потужності (оскільки є інтерпретованою мовою, є повільнішою за компільовані). Дана мова використана в рамках даної роботи як основна мова розробки програмної реалізації системи. Для даної роботи були використані наступні програмні бібліотеки (як вбудовані так і додаткові):

Таблиця 4.1 – Python-бібліотеки, використані в розробці системи

Назва	Опис
flask	Бібліотека-мікрофреймворк для побудова невеликих веб-сервісів; є основою програмної реалізації
flask-restful	Бібліотека для створення прикладних програмних інтерфейсів на основі *flask*
flask-sqlalchemy	Бібліотека, що імплементує об'єктно-реляційне відображення реляційної бази даних, з якою працює система, та програмної реалізації, за допомогою об'єктно-орієнтованого підходу
telethon	Бібліотека, що реалізує інтерфейс взаємодії системи та комунікаційної системи Telegram
jellyfish	Бібліотека, що реалізує алгоритми порівняння текстових рядків; використана для використання алгоритму Джаро-Вінклера в системі
scikit-learn	Бібліотека для обробки природньої мови та машинного навчання; використовується як джерело алгоритмів наївного Баєса та дерева прийняття рішень в системі
voluptuous	Бібліотека для контролю за форматом дозволених масивів даних
jsonschema	Бібліотека, проводити тестування вхідних даних до програми від користувачів на відповідність змісту та структури даних відповідно до контракту прикладного програмного інтерфейсу
re	Бібліотека для використання регулярних виразів при обробці текстів

Продовження таблиці 4.1

Назва	Опис
dataclasses	Бібліотека для полегшеного створення класів та надання їм додаткового функціоналу
typing	Бібліотека для анотації типів об'єктів
os	Бібліотека для виклику функцій операційної системи, взаємодії з файлами та даними безпосередньо
datetime	Бібліотека для роботи з часом та датами

4.2 Вимоги до технічного забезпечення

4.2.1 Загальні вимоги

Для коректного функціонування системи необхідно:

- операційна система на базі Linux або Unix;
- встановлена мова програмування Python 3.7.7 та вище;
- за необхідності розгортання системи як контейнера, рекомендується використовувати Docker версії 19.03.7 та вище;
- мінімальна тактова частота процесора – 1 ГГц;
- мінімальна кількість оперативної пам'яті – 2 Гб;
- мінімальна кількість пам'яті на жорсткому диску – 1 Гб.

Так як система представляє собою веб-сервіс, вимоги до засобів вводу-виводу не ставляться.

Початкова реалізація системи створена для запуску та взаємодії на локальній машині, за подальшої розробки вимоги до обладнання можуть змінюватись.

4.2.2 Опис локальної обчислювальної мережі

Система не використовує протоколи розподіленого виконання такі як MPI, локальна мережа не потрібна.

4.3 Архітектура програмного забезпечення

4.3.1 Діаграма класів

Для правильного розв'язання проблеми, що стоїть перед нами, спроектуємо діаграму класів проблемної області на рівні абстракції достатньому, щоб зобразити відношення між основними компонентами програми. Таким чином, в діаграму будуть включені ті класи, методи та атрибути, що представляють логічне представлення задачі та не залежать від конкретної реалізації. Діаграма представлена в графічних матеріалах. Опис атрибутів даних класів приведений в таблиці 4.2.

Таблиця 4.2 – Опис класів програми

Клас	Атрибут	Тип	Опис
Message	id	Ціле число	Унікальний ідентифікатор об'єкта повідомлення в системі
	text	Рядок символів	Текстовий зміст повідомлення
	creation_time	Об'єкт, представлення дати й часу	Час та дата створення повідомлення

Продовження таблиці 4.2

Клас	Атрибут	Тип	Опис
Message	channel_id	Рядок символів	Назва спільноти, звідки було агреговане повідомлення
	messenger	Рядок символів	Назва комунікаційної системи (месенджера), звідки було взяте повідомлення
Person	id	Ціле число	Унікальний ідентифікатор об'єкта особи в системі
	first_name	Рядок символів	Ім'я особи
	second_name	Рядок символів	Ім'я по-батькові особи
	last_name	Рядок символів	Прізвище особи
	jobs	Масив об'єктів класу PersonJob	Список усіх місць роботи (викладання) особи
PersonJob	faculty	Рядок символів	Назва факультету, на якому викладала особа
	cathedra	Рядок символів	Назва кафедри, де викладала особа

Продовження таблиці 4.2

Клас	Атрибут	Тип	Опис
PersonJob	subject	Рядок символів	Назва предмету, який викладала особа
Classifier	people	Колекція об'єктів класу Person	Усі особи, які потенційно можуть бути ідентифіковані в повідомленні
	message	Об'єкт типу Message	Повідомлення, за яким повинна бути проведена класифікація
BayesianClassifier	-	-	-
DecisionTreeClassifier	-	-	-

Складовими діаграми класів предметної області є:

- *Person* (Особа) – представляє собою дані про конкретну особу, яка може бути визначена в документі;
- *PersonJob* (Робота) – є представленням місця викладання / роботи конкретної особи. Ці дані будуть використані при аналізі осіб з однаковими іменами, іменами по-батькові, прізвищами після початкової обробки документів;
- *Message* (Повідомлення) – представляє собою конкретне повідомлення, агреговане з комунікаційних систем. Містить як текст, так і метаінформацію про повідомлення;
- *MessengerClientAPI* (Інтерфейс комунікаційної системи) – абстрактний клас, є узагальненням для класів, що взаємодіють з комунікаційними системами та агрегують звідти відгуки;

- *TelegramClientAPI* (Інтерфейс комунікаційної системи Telegram) – реалізація класу *MessengerClientAPI*, слугує для агрегації повідомлень з Telegram;
- *Classifier* (Класифікатор) – абстрактний клас, слугує як узагальнення для конкретних реалізацій класифікаторів осіб після початкової обробки повідомлення та формування груп осіб. Утворює з конкретними реалізаціями спрощену версію шаблону проектування "Шаблон";
- *BayesianClassifier* (Баєсівський класифікатор) – реалізація абстрактного класу *Classifier*, слугує для створення та використання програмної реалізації Баєсівського класифікатора;
- *DecisionTreeClassifier* (Класифікатор дерева прийняття рішень) – реалізація класу *Classifier*, слугує для створення та використання програмної реалізації класифікатора дерева прийняття рішень.

4.3.2 Діаграма послідовності

Для основного процесу діяльності акторів в системі створимо діаграму послідовності, що зображає взаємодію Користувача та програми, а саме основних її класів, що представляють предметну область. Зауважимо, що Користувач на даній діаграмі – виконання програмної частини, запущеної користувачем через визначений інтерфейс роботи з програмою. Діаграма представлена в графічних матеріалах.

Спочатку Користувач (програмна частина, яка відповідає за прийняття та обробку запитів від Користувача), здійснює запити для агрегації відгуків з усіх комунікаційних систем за допомогою усіх реалізацій класу *MessengerClientAPI*, викликаючи метод *get_messages()*. Кожна така реалізація на запит формує множину повідомлень, що були агреговані відповідно до параметрів запиту. Отримавши відповіді окремо від кожного виклику,

Користувач формує загальну множину повідомлень, яку далі може направити на аналіз.

Далі Користувач для кожного повідомлення визначає множину осіб, які потенційно в ньому згадані (варто згадати, що Користувач в контексті даної діаграми – програмна обробка запиту Користувача-актора) та викликає виконання методу *process()* реалізацій класу *Classifier*, які повертають йому множину осіб, найімовірніше згаданих у документі.

4.3.3 Діаграма компонентів

Для розуміння представлення системи та зовнішніх компонентів, з якими вона взаємодіє, створимо діаграму компонентів. Діаграма наведена в графічних матеріалах.

Наведемо опис компонентів, представлених на діаграмі:

- *Користувач* – представлення Користувача-актора у вигляді зовнішнього інтерфейсу, що використовує наданий системою інтерфейс взаємодії;
- *API interface* – компонент в системі, відповідальний за взаємодію системи з Користувачем, обробкою та інтерпретацією запитів Користувача. Також даний компонент взаємодіє з базою даних для управління збереженими даними. Надає *Користувачу* інтерфейс взаємодії з системою. Використовує наданий базою даних інтерфейс.
- *MessengerClientAPI* – компонент, відповідальний за взаємодію системи з комунікаційними системами. Конкретна реалізація є підгрунтям для визначення конкретної реалізації компонента *API interface*;
- *Classifier* – компонент, відповідальний за аналіз та класифікацію повідомлень, агрегованих з комунікаційних систем. Від його конкретної реалізації залежить визначення реалізації компонента *API interface*.

4.3.4 Специфікація функцій

Опишемо *усі* функції та методи програмної реалізації. Зазначимо, що для розробки програмної частини була використана мова програмування високого рівня Python, що дозволяє створювати програмні реалізації використовуючи як об'єктно-орієнтований, так і процедурний підхід водночас. Таким чином, виділяємо методи – програмні функції, що належать об'єкту певного класу та "чисті" функції, що не належать жодному з класів та можуть бути викликані без попереднього створення будь-яких попередніх об'єктів. Опис специфікації функцій наведено в таблиці 4.3.

Таблиця 4.3 – Опис функцій та методів програмної реалізації

Клас	Функція (метод)	Опис
MessengerClient API (абстрактний)	get_messages	Агрегування повідомлень з комунікаційної системи (месенджера)
	accepts	Перевірка, чи реалізація може прийняти запит на агрегування повідомлень відповідно до даних, що надав Користувач
	validation_schema	-
TelegramClient API	get_messages	Агрегування повідомлень з месенджера Telegram
	_get_messages	Функція для виклику об'єкту бібліотеки Telethon для агрегації повідомлень
	validation_schema	-
	accepts	Перевірка, чи запит на агрегацію з Telegram
Classifier	process	Обробка повідомлень та визначення згаданих у них людей
	get_core_classifier	Створити та підготувати об'єкт бібліотеки scikit-learn для класифікації

Продовження таблиці 4.3

Клас	Функція (метод)	Опис
Classifier	find_maximum_proba_index	Знайти індекс людини, що найімовірніше з групи згадана у повідомленні
	group_people_by_name	Групувати людей за ім'ям, ім'ям по-батькові та прізвищем
BayesianClassifier	get_core_classifier	Створити об'єкт Баєсівського класифікатора
DecisionTreeClassifier	get_core_classifier	Створити об'єкт дерева прийняття рішень
Person	as_tuple	Серіалізувати об'єкт у формат для використання його як ключа для хеш-функції
	from_raw_data	Створити об'єкт, використовуючи вхідні дані від Користувача
	validation_schema	-
PersonJob	from_raw_data	Створити об'єкт, використовуючи вхідні дані від Користувача
	validation_schema	-
SubstitutionModel	substitutions	Десеріалізувати асоційовані теги з бази даних з рядка символів у об'єкт
	substitutions	Серіалізувати асоційовані теги з об'єкта в рядок символів для запису в базу даних
	from_raw_substitutions	Створити об'єкт, використовуючи вхідні дані від Користувача

Продовження таблиці 4.3

Клас	Функція (метод)	Опис
ApplicationConfig	db_url	Створити посилання для доступу до бази даних зі змінних оточення
SubstitutionManagement	get	Зчитати з бази даних і відправити Користувачу дані про асоційовані теги
	post	Записати в базу даних асоційований тег
	delete	Видалити з бази даних асоційований тег
	_post_validation_schema	Схема для перевірки правильності формату даних, надісланих Користувачем
Analysis	post	Запустити аналіз відгуків та повернути Користувачу результати аналізу
	_post_validation_schema	Схема для перевірки правильності формату даних, надісланих Користувачем
Encoded Data	_flatten_data	Агрегувати дані про місця роботи по усіх особах
	vectorized_data	Перетворити агреговані дані на відповідні теги
Message Analyzer	all_tags_groups	Отримати усі можливі теги, згадані у повідомленні (факультети, кафедри, назви дисциплін)
	_find_by_short_info	Знайти тег у повідомленні за скороченням (абревіатурою)
	_find_faculties_in_message	Знайти назви факультетів у повідомленні
	_find_cathedras_in_message	Знайти назви кафедр у повідомленні

Продовження таблиці 4.3

Клас	Функція (метод)	Опис
Message Analyzer	_find_subjects_in_message	Знайти назви дисциплін у повідомленні
	_shorten_tag	Скоротити назву тегу до абревіатури
ValidationSchema (абстрактний)	validation_schema	Створення об'єкту для перевірки правильності формату й змісту даних, наданих Користувачем
-	get_message_person_mapping	Сформувати список повідомлень згаданих у них людей
-	process_all_people_per_message	Сформувати список людей, згаданих у повідомленні
-	filter_people_by_passport_info	Сформувати список людей, згаданих у повідомленні, за ім'ям, прізвищем, ім'ям по-батькові

4.4 Опис звітів

Структура звіту результатів аналізу, сформованого за даними та на запит Користувача, описано в контракті прикладного програмного інтерфейсу програми та Користувача, наведеного в Додатку Б.

Висновок до розділу

В даному розділі були розглянуті інструменти, технології, засоби розробки програмних продуктів, використаних під час створення програмної реалізації системи.

Був приведений список та короткий опис інструментів, використаних під час проектування, аналізу та створення програмного продукту. Були

визначена система управління базами даних, а також основна мова програмування розробки, наведений короткий опис та призначення програмних бібліотек, що використовуються.

Були визначені апаратні та програмні вимоги до технічного забезпечення для розгортання програмного продукту.

Була приведена діаграма класів програмної розробки, що повинні бути імплементовані в системі незалежно від конкретної програмної реалізації. Була приведена діаграма послідовності взаємодії описаних класів. Створена діаграма компонентів, що представляє визначає фізичну побудову системи. Був наведений опис усіх функцій програмної реалізації на мові програмування Python.

5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1 Керівництво користувача

Під час розробки системи був створений програмний продукт, що вирішує поставлені перед нами задачі агрегації, аналізу та класифікації відгуків з комунікаційних мереж. Даний програмний продукт був виконаний у вигляді веб-сервісу *без* візуального інтерфейсу Користувача. Користувачем для розробленої системи вважається стороння автоматизована система, що здійснює відповідні запити з використанням прикладного програмного інтерфейсу, обґрунтування, вхідні та вихідні дані якого наведені в розділі 2 даної роботи.

Користувач, згідно розділу 1.1.2, може виконувати наступні функції:

- створення асоційованих тегів;
- видалення асоційованих тегів;
- отримання інформації про усі асоційовані теги, наявні у базі даних;
- запуск агрегації та аналізу відгуків.

Для кожної з даних функцій була відповідні програмні складові для забезпечення можливості використання системи Користувачем.

Зобразимо взаємодію Користувача та системи за допомогою програмного забезпечення Postman, що дозволить візуально зобразити дану взаємодію у вигляді запитів та відповідей протоколу HTTPS.

У випадку, якщо Користувач для *будь-якого* запиту надає дані у форматі та складі, не визначеному контрактом прикладного програмного забезпечення, система повертає повідомлення, зображене на рис. 5.1 (наведений приклад запиту неправильного формату).

Рисунок 5.1 - Відповідь системи на неправильний формат запиту

Користувач може створити новий тег та список асоційованих з ним значень для використання даних значень під час подальших аналізів. Запит на створення нового тегу та відповідь системи приведені на рис. 5.2.

Рисунок 5.2 - Запит на створення тегу та відповідь системи

Користувач може видалити існуючий тег та асоційовані з ним значення з бази даних. Приклад такого запиту та відповіді про успішне видалення наведено на рис. 5.3.

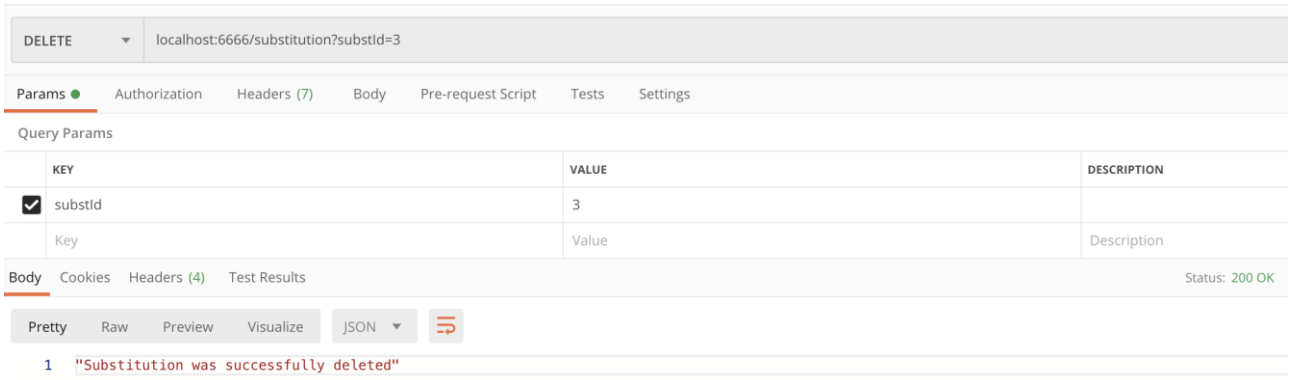


Рисунок 5.3 - Запит на видалення тегу та відповідь системи

У випадку, якщо тегу з вказаним ідентифікатором не було знайдено, система поверне повідомлення про відсутній тег та відповідний код відповіді, показані на рис. 5.4.

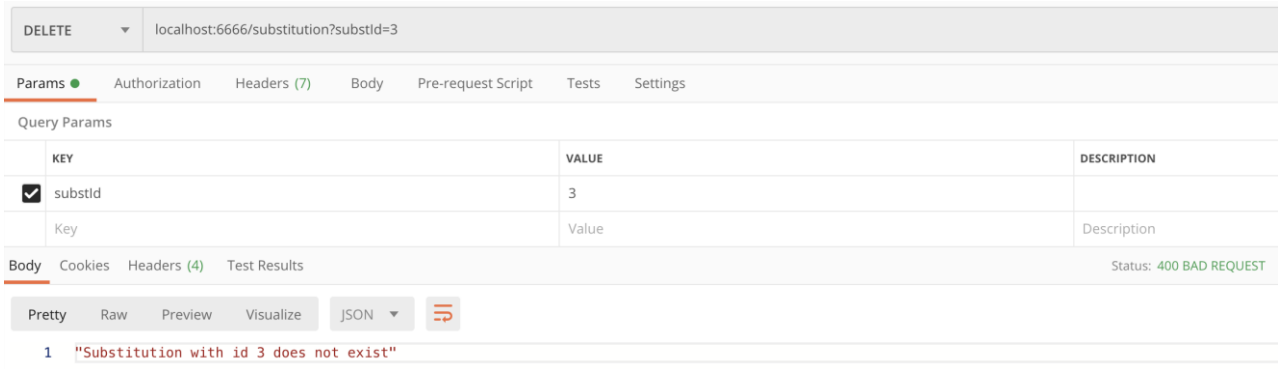


Рисунок 5.4 - Запит на видалення тегу з неіснуючим ідентифікатором та відповідь системи

Користувач має можливість отримати інформацію про усі збережені у базі теги та асоційовані значення. Приклад запиту на отримання інформації та відповідь системи показані на рис. 5.5.

Рисунок 5.5 - Запит на отримання інформації про усі теги та відповідь системи

Користувач має можливість запустити агрегацію та подальший аналіз відгуків з сторонніх комунікаційних систем. Приклад такого запиту зображений на рис. 5.6.

Рисунок 5.6 - Приклад запиту на проведення аналізу

У відповідь на даний запит, система повертає відповідь у форматі application/json дані, визначені у контракті прикладного програмного інтерфейсу. Приклад запиту та відповіді приведені на рис. 5.7.

Рисунок 5.7 - Відповідь системи на запит про проведення аналізу

5.2 Випробування програмного продукту

5.2.1 Мета випробувань

Метою випробувань являється перевірка відповідності функцій комплексу задач класифікації анонімних відгуків стосовно осіб, що згадані у них, вимогам технічного завдання; правильність обробки вхідних даних від Користувача.

5.2.2 Загальні положення

Випробування проводяться на основі наступних документів:

– ГОСТ 34.603–92. Інформаційна технологія. Види випробувань автоматизованих систем.
– ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

5.2.3 Результати випробувань

Проведемо ручне тестування повноту функціональної моделі, визначеної під час проектування системи, тестування функціональних вимог та тестування точності класифікації створеної системи. Тестування проводиться в ручному режимі, так як можемо використати засоби для взаємодії системи за допомогою прикладного програмного інтерфейсу, а саме Postman.

Проведемо випробування на повноту функціональної моделі. Результати даних випробувань представлені в таблиці 5.1.

Таблиця 5.1 – Випробування системи на функціональну повноту

Передумова	Дія	Кроки тесту	Післяумова	Результат тесту
Система розгорнута та приймає запити від користувача	Створити асоційований тег	Надіслати запит на створення тегу	Надісланий тег був записаний до бази даних	Пройдений
	Видалити асоційований тег	Надіслати запит на видалення тегу	Тег з вказаним у запиті ідентифікатором був видалений	Пройдений
	Отримати інформацію про асоційовані теги	Надіслати запит на отримання інформації	Була отримана інформація про *усі* записані до бази даних асоційованих тегів	Пройдений
	Запустити аналіз відгуків	Надіслати запит на проведення аналізу	Була отримана інформація про агреговані повідомлення та проаналізованих осіб	Пройдений

Проведемо тестування функціональних вимог, визначених в технічному завданні. Результати даного тестування наведені в таблиці 5.2.

Таблиця 5.2 – Випробування системи на відповідність вимогам

Передумова	№ з/п вимоги	Кроки тесту	Післяумова	Результат тесту
Система розгорнута та приймає запити від користувача	1	Надіслати запит на проведення аналізу з різними джерелами даних	Були агреговані повідомлення з *усіх* вказаних у запиті джерел	Пройдений
	2	Надіслати запит на проведення аналізу, включивши інформацію хоча би про одну особу	Система зберегла отримані дані про особу в жодному вигляді	Пройдений
	3	Надіслати запит на проведення аналізу з двома особами з абсолютно ідентичною інформацією, але різними ідентифікаторами; дані такі, щоб особи гарантовано були розпізнані у відгуку	Система не позначила жодну з осіб як отримувача відгуку	Пройдений

Продовження таблиці 5.2

Передумова	№ з/п вимоги	Кроки тесту	Післяумова	Результат тесту
Система розгорнута та приймає запити від користувача	4	Надіслати запит, сформувавши масив даних з некоректним форматом	Система повернула повідомлення про неправильність формату	Пройдений

Додатково проведемо тестування системи на кінцеву точність класифікації. Для проведення даного випробування був створений тестовий набір даних, що включає у себе:

- 130 повідомлень українською мовою;
- 87 повідомлень російською мовою;
- 113 даних про осіб.

Набір даних був створений на основі каналів комунікаційної системи Telegram з ідентифікаторами *kpi_777* та *kpi_666*. Після початкового збирання даних була проведена санітизація для уникнення проблем з використанням інформації, зібраної про осіб з відкритих джерел (офіційні сайти кафедр та факультетів НТУУ "КПІ ім. Ігоря Сікорського").

Була проведена точність виділення груп осіб у текстах шляхом підрахунку кількості правильних результатів виділення на основі відгуків українською та російською мовами. Результати даного випробування наведені на рис. 5.8.

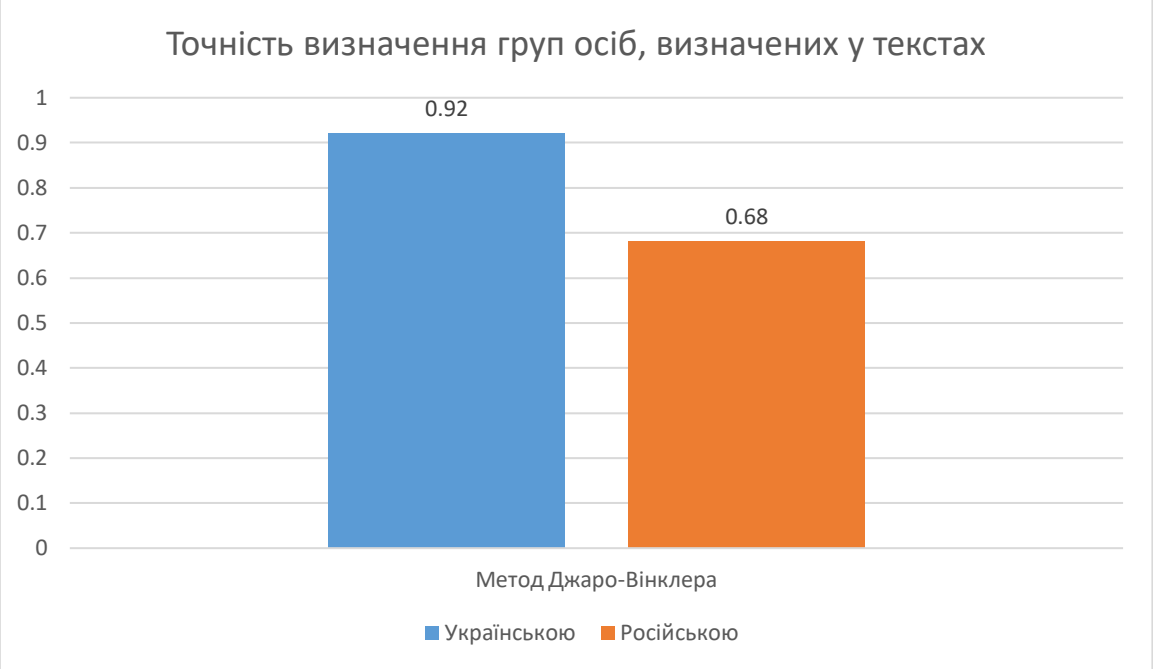


Рисунок 5.8 – Результати визначення груп осіб у текстах

Далі було проведене дослідження визначення конкретних осіб з груп осіб, знайдених у текстах. Для даного випробування була проведена не тільки визначення точності, а ще й повноти. Точність у даному випробуванні показуватиме нам, яка відносна кількість осіб була класифікована, тоді як повнота – кількість осіб, що була класифікована правильно з множини взагалі класифікованих. Дані випробувані проводились на навчальних даних з українською мовою. Результати випробування представлені на рис. 5.9.

Рисунок 5.9 – Точність та повнота визначення особи з групи

Висновок до розділу

У даному розділі була визначена мета проведення тестування розробленої системи. Було проведене тестування системи відповідно до функціональних варіантів використання та визначені кроки для кожного з них. Було проведене тестування на відповідність системи функціональним вимогам, визначеним в технічному завданні до розробки системи.

ЗАГАЛЬНІ ВИСНОВКИ

Завдяки впровадженню інформаційних технологій в навчальний процес вдається досягти нових можливостей взаємодії викладача та студента. Важливим аспектом такої взаємодії є можливість залишити відгук з суб'єктивною оцінкою про якість викладання предмету для можливого пошуку та виправлень прогалин в навчальному процесі. Завдяки розповсюдженню комунікаційних систем, сьогодні все більше людей об'єднуються в тематичні спільноти, де анонімно діляться відгуками про навчальний процес з широким колом людей. З даних причин постає завдання збору та аналізу даних, що мають неструктуровану форму, на предмет позначених у них осіб. Ручне тестування витрачає багато ресурсів, в той час автоматизовані системи могли б легко масштабувати збір та аналіз відгуків без збільшення вартості проведення даних операцій. Розробка такої системи – мета даної дипломної роботи.

Спочатку був описаний та визначений процес діяльності, необхідний для забезпечення системою, що розроблялась, виконання своїх функцій. Була сформована мета розробки та завдання, що мають бути вирішені.

Далі була визначена структура вхідних та вихідних даних системи, а також структура даних, які система зберігає до сховищ даних (база даних, файлова система тощо).

Були визначені змістовна та математична постановка задачі, наведені та описані методи їх розв'язання.

Були описані технології та інструменти, за допомогою яких була створена система. Описані вимоги для розгортання системи на електронно-обчислювальних машинах. Була визначена програмна структура системи, що описує проблему, що поставлена перед нами. Визначена послідовність виконання компонентів системи під час основної діяльності.

Було визначено мету випробувань та проведене тестування системи на повноту визначеної функціональної моделі, а також функціональних вимог.

ПЕРЕЛІК ПОСИЛАНЬ

1. Закон України "Про освіту" : за станом на 5 вер. 2017 р. / Верховна Рада України. – Офіц. вид. – К. : Голос України, 2017.
2. Cambridge Dictionary [Електронний ресурс] – Режим доступу: https://dictionary.cambridge.org.
3. No Free Lunch Theorems for Optimization / David H. Wolpert, William G. Macready // IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION. – 1997, Vol.1. – p. 67.
4. spaCy: Facts & Figures [Електронний ресурс] – Режим доступу: https://spacy.io/usage/facts-figures.
5. David H. Wolpert. Natural Language Processing with Python / David H. Wolpert, William G. Macready. : O'REILLY, 2006.
6. The Stanford CoreNLP Natural Language Processing Toolkit / Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, David McClosky // The Stanford Natural Language Processing Group. – Stanford, 2014.
7. OpenAPI Specification [Електронний ресурс] – Режим доступу: http://spec.openapis.org/oas/v3.0.3.
8. YAML: YAML Ain't Markup Language [Електронний ресурс] – Режим доступу: https://yaml.org.
9. J. Quinlan. Induction of Decision Trees / J. Quinlan. – Boston, Kluwer Academics Publishers, 1986.
10. Cardinal Number [Електронний ресурс] – Режим доступу: https://mathworld.wolfram.com/CardinalNumber.html.
11. T. Bergmanis. Context Sensitive Neural Lemmatization with Lematus / Toms Bergmanis, Sharon Goldwater // Association for Computational Linguistics. – 2018, - p. 1391-1392.

12. A Comparison of String Distance Metrics for Name-Matching Tasks / William W. Cohen, Pradeep Ravikumar, Stephen E. Fienberg. – Pittsburg, 2003. – Carnegie Mellon University.
13. String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage / William E. Winkler. – 1990. – U.S. Census bureau.
14. Christopher D. Manning. Introduction to Information Retrieval / Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze. – 2008. – Cambridge University Press.
15. Lior Rokach. DATA MINING WITH DECISION TREES. Theory and Applications, 2nd edition / Lior Rokach, Oded Maimon. – Singapore, 2015.
16. macOS Catalina [Електронний ресурс] – Режим доступу: https://www.apple.com/macos/catalina.
17. PyCharm: The Python IDE for Professional Developers [Електронний ресурс] – Режим доступу: https://www.jetbrains.com/pycharm.
18. Visual Studio Code [Електронний ресурс] – Режим доступу: https://code.visualstudio.com.
19. Diagrams in Confluence and JIRA [Електронний ресурс] – Режим доступу: https://drawio-app.com.
20. Postman: The Collaboration Platform for API Development [Електронний ресурс] – Режим доступу: https://www.postman.com.
21. Git [Електронний ресурс] – Режим доступу: https://git-scm.com.
22. PostgreSQL: The World's Most Advanced Open Source Relational Database [Електронний ресурс] – Режим доступу: https://www.postgresql.org.
23. Docker [Електронний ресурс] – Режим доступу: https://www.docker.com.

24. GNU Make [Електронний ресурс] – Режим доступу: https://www.gnu.org/software/make.
25. Python Programming Language [Електронний ресурс] – Режим доступу: https://www.python.org.

Додаток А

Тексти програмного коду

Інформаційна система аналізу персоналізованих відгуків учасників навчального процесу

(Найменування програми (документа))

DVD-R

(Вид носія даних)

30 арк, 72 Кб

(Обсяг програми (документа) , арк.,) Кб)

Київ – 2020 року

main.py

```
import configure_app  # noqa
from flask_app import app

# TODO :: would be nice to make some logging?

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=8050)
```

flask_db.py

```
from flask_sqlalchemy import SQLAlchemy

from application_config import APPLICATION_CONFIG
from flask_app import app

app.config['SQLALCHEMY_DATABASE_URI'] = APPLICATION_CONFIG.db_url
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)
```

flask_app.py

```
from flask import Flask

app = Flask(__name__)
```

configure_app.py

```
from flask_restful import Api

import models  # noqa
from api.analysis import Analysis
from api.substitution import SubstitutionManagement
```

```
from flask_app import app
from flask_db import db


api = Api(app)


api.add_resource(Analysis, '/analysis')
api.add_resource(SubstitutionManagement, '/substitution')


db.create_all()
db.session.commit()
```

application_config.py

```
import os
from dataclasses import dataclass


from voluptuous import Schema, REMOVE_EXTRA, Optional


SCHEMA = Schema({
    Optional('DB_USER', default='user'): str,
    Optional('DB_HOST', default='localhost'): str,
    Optional('DB_PORT', default='5433'): str,
    Optional('DB_PASS', default='password'): str,
    Optional('DB_NAME', default='analyzerdb'): str,
    Optional('TELEGRAM_API_ID', default=''): str,
    Optional('TELEGRAM_API_KEY', default=''): str,
    Optional('TELEGRAM_API_STRING', default=''): str
}, extra=REMOVE_EXTRA)



@dataclass(frozen=True)
```

```
class ApplicationConfig:
    DB_USER: str
    DB_HOST: str
    DB_PORT: str
    DB_PASS: str
    DB_NAME: str
    TELEGRAM_API_ID: str
    TELEGRAM_API_KEY: str
    TELEGRAM_API_STRING: str


    @property
    def db_url(self) -> str:
        return
f'postgres://{self.DB_USER}:{self.DB_PASS}@{self.DB_HOST}:{self.DB_PORT}/{self.DB_
NAME}'



APPLICATION_CONFIG = ApplicationConfig(**SCHEMA({
    key: value for key, value in os.environ.items()
}))
```

api/analysis.py

```
from typing import List, Dict, Any


import jsonschema
from flask import request
from flask_restful import Resource


from core.messengers_clients.telegram import TelegramClientAPI
from core.common import Person, Message
from message_processing.message_processor import get_message_person_mapping
```

```
class ApiClients:
    telegram = 'telegram'



API_CLIENT_BY_NAME_MAPPING = {
    ApiClients.telegram: TelegramClientAPI
}



def analysis_result(all_messages: List[Message], filtered_people: Dict[Message, List[Person]]) -
> List[Dict]:
    prepared_data: List[Dict] = []
    for message in all_messages:
        message_info = {
            'id': message.id,
            'text': message.text,
            'creation_time': message.creation_time.isoformat(),
            'channel_id': message.channel_id,
            'messenger': message.messenger,
            'mentioned_people': [person.id for person in filtered_people[message]]
        }
        prepared_data.append(message_info)
    return prepared_data



class Analysis(Resource):


    def post(self):
```

```
    data: Dict = request.get_json(silent=True)
    try:
        jsonschema.validate(data, self._post_validation_schema())
    except jsonschema.ValidationError:
        return 'Input values are incorrect', 400

    people = [Person.from_raw_data(person_data) for person_data in data['people']]
    messages: List[Message] = []
    for source_data in data['sources']:
        for client_class in API_CLIENT_BY_NAME_MAPPING.values():
            if client_class.accepts(**source_data):
                client = client_class(**source_data)
                messages.extend(client.get_messages())
    filtered_people = get_message_person_mapping(people, messages)
    prepared_data = analysis_result(messages, filtered_people)
    return prepared_data, 200

def _post_validation_schema(self) -> Any:
    return {
        'type': 'object',
        'properties': {
            'people': {
                'type': 'array',
                'items': Person.validation_schema()
            },
            'sources': {
                'type': 'array',
                'items': {
                    'anyOf': [
```

```
                    client_class.validation_schema() for client_class in
API_CLIENT_BY_NAME_MAPPING.values()
                ]
            }
        }
    },
    'additionalProperties': True,
    'required': ['people', 'sources']
}
```

api/substitution.py

```
from typing import Dict, Optional, Any, List

import jsonschema
from flask import request
from flask_restful import Resource
from flask_restful.reqparse import RequestParser

from flask_db import db
from models import SubstitutionModel



class SubstitutionManagement(Resource):

    def __init__(self, *args, **kwargs):
        self.reqparse = RequestParser()
        self.reqparse.add_argument('substId', type=int, required=True)
        super().__init__(*args, **kwargs)
```

```
def get(self):
    substs: List[SubstitutionModel] = db.session.query(SubstitutionModel).all()
    dumped_data = [{
        'id': subst.id,
        'target': subst.target,
        'substitutions': subst.substitutions
    } for subst in substs]
    return dumped_data, 200


def post(self):
    data: Dict = request.get_json(silent=True)
    try:
        jsonschema.validate(data, self._post_schema())
    except jsonschema.ValidationError:
        return 'Input values are incorrect', 422
    subst_data: Dict = data['substitution_data']
    target = subst_data['target']
    substs = subst_data['substitutions']
    subst_obj: List[SubstitutionModel] = db.session.query(SubstitutionModel).filter(
        SubstitutionModel.target == target
    ).one_or_none()
    if subst_obj is None:
        subst_obj = SubstitutionModel.from_raw_substitutions(target, substs)
    else:
        subst_obj.substitutions = substs
    db.session.add(subst_obj)
    db.session.commit()
    return 'Substitution successfully set', 201


def _post_schema(self) -> Any:
```

```
        return {
            'type': 'object',
            'properties': {
                'substitution_data': {
                    'type': 'object',
                    'properties': {
                        'target': {
                            'type': 'string',
                            'minLength': 1
                        },
                        'substitutions': {
                            'type': 'array',
                            'items': {
                                'type': 'string',
                                'minLength': 1
                            }
                        }
                    },
                    'additionalProperties': False,
                    'required': ['target', 'substitutions']
                }
            },
            'additionalProperties': True,
            'required': ['substitution_data']
        }

    def delete(self):
        args = self.reqparse.parse_args()
        subst_id = args['substId']
        subst_obj: Optional[SubstitutionModel] = db.session.query(SubstitutionModel).filter(
```

```
            SubstitutionModel.id == subst_id
        ).one_or_none()
        if subst_obj is None:
            return f'Substitution with id {subst_id} does not exist', 400
        db.session.delete(subst_obj)
        db.session.commit()
        return 'Substitution was successfully deleted', 200
```

cli/__main__.py

```
import click

from cli.telegram_session_generation import generate_telegram_session_string



@click.group()
def register_cli():
    pass



register_cli.add_command(generate_telegram_session_string)

if __name__ == '__main__':
    register_cli()
```

cli/telegram_session_generation.py

```
import click
from telethon import TelegramClient
from telethon.sessions import StringSession
```

```
@click.command(
    name='generate-telegram-session-string',
    help='Prints generated session to avoid repeating of messages sending'
)
@click.option('-i', '--auth_id', type=int, required=True)
@click.option('-k', '--auth_key', type=str, required=True)
def generate_telegram_session_string(auth_id: int, auth_key: str):
    with TelegramClient(StringSession(), auth_id, auth_key) as client:
        print(f'Session string for Telegram authentication: {client.session.save()}')
```

core/classification/bayesian.py

```
from sklearn.naive_bayes import CategoricalNB

from core.classification.classifier import Classifier
from core.classification.common import EncodedData



class BayesianClassifier(Classifier):
    def get_core_classifier(self, encoded_data: EncodedData) -> CategoricalNB:
        class_priors = [0] + [1] * (len(encoded_data.ids_encoder.classes_) - 1)
        return CategoricalNB(class_prior=class_priors)
```

core/classification/classifier.py

```
from abc import ABCMeta, abstractmethod
from collections import defaultdict
from math import isclose
from typing import List, Tuple, Dict, Optional

import numpy as np
```

```
from core.classification.common import EncodedData, MessageAnalyzer
from core.common import Person, Message
from flask_db import db
from models import SubstitutionModel



class Classifier(metaclass=ABCMeta):
    def __init__(self, people: List[Person], message: Message) -> None:
        self.people = people
        self.message = message


    @staticmethod
    def find_maximum_proba_index(predictions: np.ndarray) -> Optional[float]:
        """Calculate index of a row with maximum value"""
        aggregated_probas: np.array = predictions.sum(axis=0)
        sorted_probas: np.array = np.sort(aggregated_probas)
        # If 2 floats are equal with enough tolerance, we can say there are more than 1 person
determined
        if isclose(sorted_probas[-1], sorted_probas[-2], abs_tol=1e-8):
            return
        return aggregated_probas.argmax()


    def process(self) -> List[Person]:
        person_from_each_group: List[Person] = []
        people_groups = self.group_people_by_name()
        substitutions: List[SubstitutionModel] = db.session.query(SubstitutionModel).all()
        for people_group in people_groups.values():
            if len(people_group) == 1:
                person_from_each_group.append(people_group[0])
                continue
```

```
            encoded_data = EncodedData(people_group)
            classifier = self.get_core_classifier(encoded_data)
            classifier.fit(*encoded_data.vectorized_data)
            message_analyzer = MessageAnalyzer(self.message, encoded_data, substitutions)
            all_tags_grouped = message_analyzer.all_tags_groups()
            if not all_tags_grouped:
                continue
            # Calculate all prediction probabilities
            prediction: np.ndarray = classifier.predict_proba(all_tags_grouped)
            max_index = self.find_maximum_proba_index(prediction)
            if max_index:
                decoded_id: int = int(encoded_data.ids_encoder.inverse_transform([max_index])[0])
                person_from_each_group.append(encoded_data.person_by_id[decoded_id])
        return person_from_each_group

    def group_people_by_name(self) -> Dict[Tuple[str, ...], List[Person]]:
        grouped_people: Dict[Tuple[str, ...], List[Person]] = defaultdict(list)
        for person in self.people:
            grouped_people[person.as_tuple()].append(person)
        return grouped_people

    @abstractmethod
    def get_core_classifier(self, encoded_data: EncodedData):
        raise NotImplementedError()
```

core/classification/common.py

```
import re
from typing import List, Dict, Tuple

import numpy as np
```

```
from sklearn.preprocessing import LabelEncoder

from core.classification.semantical_data import STOP_WORDS
from core.common import Person, Message, SINGLE_WORD
from models import SubstitutionModel

EMPTY_DATA_FILLER = ''



class EncodedData:
    """Class that is responsible for processing the encoded"""

    def __init__(self, people: List[Person]) -> None:
        self._people = people
        self.person_by_id: Dict[int, Person] = {person.id: person for person in people}

        self.flat_people_data = self._flatten_data()
        all_faculties, all_cathedras, all_subjects, all_ids = self.flat_people_data

        self.faculties_encoder = LabelEncoder().fit(all_faculties)
        self.cathedras_encoder = LabelEncoder().fit(all_cathedras)
        self.subjects_encoder = LabelEncoder().fit(all_subjects)
        self.ids_encoder = LabelEncoder().fit(all_ids)

    def _flatten_data(self) -> Tuple[List, List, List, List]:
        faculties: List[str] = [EMPTY_DATA_FILLER]
        cathedras: List[str] = [EMPTY_DATA_FILLER]
        subjects: List[str] = [EMPTY_DATA_FILLER]
        ids: List[int] = [EMPTY_DATA_FILLER]
```

```
        for person in self._people:
            for job in person.jobs:
                faculties.append(job.faculty)
                cathedras.append(job.cathedra)
                subjects.append(job.subject)
                ids.append(person.id)
        return faculties, cathedras, subjects, ids


    @property
    def vectorized_data(self) -> Tuple[np.array, List[str]]:
        all_faculties, all_cathedras, all_subjects, all_ids = self.flat_people_data


        return np.column_stack((
            self.faculties_encoder.transform(all_faculties),
            self.cathedras_encoder.transform(all_cathedras),
            self.subjects_encoder.transform(all_subjects)
        )), self.ids_encoder.transform(all_ids)



class MessageAnalyzer:

    def __init__(self, message: Message, encoded_data: EncodedData, substitutions:
List[SubstitutionModel]) -> None:
        self.message = message
        self.encoded_data = encoded_data
        self.substitutions_by_target: Dict[str, SubstitutionModel] = {subst.target: subst for subst in
substitutions}


    def all_tags_groups(self) -> List[Tuple[int, int, int]]:
        found_faculties: List[str] = self._find_faculties_in_message()
        found_cathedras: List[str] = self._find_cathedras_in_message()
        found_subjects: List[str] = self._find_subjects_in_message()
```

```
        encoded_faculties: List[int] =
self.encoded_data.faculties_encoder.transform(found_faculties)

        encoded_cathedras: List[int] =
self.encoded_data.cathedras_encoder.transform(found_cathedras)

        encoded_subjects: List[int] =
self.encoded_data.subjects_encoder.transform(found_subjects)


        grouped_faculties: List[Tuple[int, int, int]] = [(faculty, 0, 0) for faculty in
encoded_faculties]

        grouped_cathedras: List[Tuple[int, int, int]] = [(0, cathedra, 0) for cathedra in
encoded_cathedras]

        grouped_subjects: List[Tuple[int, int, int]] = [(0, 0, subject) for subject in encoded_subjects]

        return grouped_faculties + grouped_cathedras + grouped_subjects


    def _find_by_short_info(self, words_matches: List[str], tags_classes: List[str]) -> List[str]:

        found_tags: List[str] = []

        for word_match in words_matches:

            for tag_class in tags_classes:

                short_tag: str = self._shorten_tag(tag_class)

                if tag_class in self.substitutions_by_target:

                    substs = self.substitutions_by_target[tag_class]

                    if short_tag in substs.substitutions:

                        tag_class_tags = substs.substitutions

                    else:

                        tag_class_tags = [short_tag] + substs.substitutions

                else:

                    tag_class_tags = [short_tag]

                for tag in tag_class_tags:

                    if tag.lower() == word_match.lower():

                        found_tags.append(tag_class)

                        continue
```

```
        return found_tags

    def _find_faculties_in_message(self) -> List[str]:
        words_matches: List[str] = re.findall(r'(?<=#)\w+', self.message.text, re.UNICODE)
        tags_classes: List[str] = self.encoded_data.faculties_encoder.classes_
        return self._find_by_short_info(words_matches, tags_classes)

    def _find_cathedras_in_message(self) -> List[str]:
        words_matches: List[str] = re.findall(r'\w+', self.message.text, re.UNICODE)
        tags_classes: List[str] = self.encoded_data.cathedras_encoder.classes_
        return self._find_by_short_info(words_matches, tags_classes)

    def _find_subjects_in_message(self) -> List[str]:
        words_matches: List[str] = re.findall(r'\w+', self.message.text, re.UNICODE)
        tags_classes: List[str] = self.encoded_data.subjects_encoder.classes_
        return self._find_by_short_info(words_matches, tags_classes)

    @staticmethod
    def _shorten_tag(tag_class: str) -> str:
        tag_words: List[str] = re.findall(SINGLE_WORD, tag_class, re.UNICODE)
        result_tag: List[str] = []
        for word in tag_words:
            if word.lower() not in STOP_WORDS:
                result_tag.append(word[0])
        return ''.join(result_tag)
```

core/classification/decision_tree.py

```
from sklearn.tree import DecisionTreeClassifier as CoreDecisionTreeClassifier
```

```
from core.classification.classifier import Classifier
from core.classification.common import EncodedData


class DecisionTreeClassifier(Classifier):
    def get_core_classifier(self, encoded_data: EncodedData) -> CoreDecisionTreeClassifier:
        return CoreDecisionTreeClassifier()
```

core/classification/semantical_data.py

```
STOP_WORDS = {
    'і',
    'й',
    'та',
    'кафедра'
}
```

core/messengers_clients/messenger_client.py

```
from abc import ABC, abstractmethod
from typing import List

from core.common import Message, ValidationSchema


class MessengerClientAPI(ValidationSchema):
    """Interface for every messenger client with minimal methods set"""

    @abstractmethod
    def get_messages(self) -> List[Message]:
        raise NotImplementedError()
```

```
    @classmethod
    @abstractmethod
    def accepts(cls, **data) -> bool:
        raise NotImplementedError()
```

core/messengers_clients/telegram.py

```
import asyncio
from datetime import datetime
from typing import List, Any

from telethon import TelegramClient
from telethon.sessions import StringSession

from application_config import APPLICATION_CONFIG
from core.common import Message
from core.messengers_clients.messenger_client import MessengerClientAPI


class TelegramClientAPI(MessengerClientAPI):
    def __init__(self, **data) -> None:
        self._api_id: int = APPLICATION_CONFIG.TELEGRAM_API_ID
        self._api_key: str = APPLICATION_CONFIG.TELEGRAM_API_KEY
        self._string_session: str = APPLICATION_CONFIG.TELEGRAM_API_STRING
        self._channels: List[str] = data['channels']
        self._start_timestamp = datetime.fromisoformat(data['start_time'])
        self._end_timestamp = datetime.fromisoformat(data['end_time'])

    async def _get_messages(self) -> List[Message]:
        messages: List[Message] = []
```

```
        async with TelegramClient(StringSession(self._string_session), self._api_id, self._api_key)
as client:
            for channel in self._channels:
                async for t_message in client.iter_messages(channel,
offset_date=self._end_timestamp):
                    if t_message.date.replace(tzinfo=None) <
self._start_timestamp.replace(tzinfo=None):
                        break
                    parsed_message = Message(
                        id=t_message.id,
                        text=t_message.message,
                        creation_time=t_message.date,
                        channel_id=channel,
                        messenger='telegram'
                    )
                    messages.append(parsed_message)
        return messages

    def get_messages(self) -> List[Message]:
        res = asyncio.run(self._get_messages())
        return res

    @classmethod
    def validation_schema(cls) -> Any:
        return {
            'type': 'object',
            'properties': {
                'id': {
                    'type': 'string',
                    'enum': ['telegram']
                },
```

```
            'channels': {
                'type': 'array',
                'items': {
                    'type': 'string'
                }
            },
            'start_time': {
                'type': 'string',
                'format': 'date-time'
            },
            'end_time': {
                'type': 'string',
                'format': 'date-time'
            }
        },
        'additionalProperties': False,
        'required': ['id', 'channels', 'start_time', 'end_time']
    }

@classmethod
def accepts(cls, **data) -> bool:
    return data['id'] == 'telegram'
```

core/common.py

```
from abc import ABC, abstractmethod
from dataclasses import dataclass
from datetime import datetime
from typing import List, Any, Tuple, Dict


SINGLE_WORD = r'\w+'
```

```
class ValidationSchema(ABC):
    @classmethod
    @abstractmethod
    def validation_schema(cls) -> Any:
        raise NotImplementedError()


@dataclass(eq=True, frozen=True)
class PersonJob(ValidationSchema):
    faculty: str
    cathedra: str
    subject: str

    @classmethod
    def from_raw_data(cls, data: Dict) -> 'PersonJob':
        return cls(
            faculty=data['faculty'],
            cathedra=data['cathedra'],
            subject=data['subject']
        )

    @classmethod
    def validation_schema(cls) -> Any:
        return {
            'type': 'object',
            'properties': {
                'faculty': {
                    'type': 'string',
```

```
                'minLength': 1
            },
            'cathedra': {
                'type': 'string',
                'minLength': 1
            },
            'subject': {
                'type': 'string',
                'minLength': 1
            }
        },
        'additionalProperties': False,
        'required': ['faculty', 'cathedra', 'subject']
    }


@dataclass(frozen=True)
class Person(ValidationSchema):
    id: int
    first_name: str
    second_name: str
    last_name: str
    jobs: List[PersonJob]

    # def __post_init__(self) -> None:
    #     if not self.jobs:
    #         raise ValueError('Person should have at least 1 job')

    def __repr__(self) -> str:
        return f'{self.last_name} ({self.id})'
```

```
def as_tuple(self) -> Tuple[str, ...]:
    return self.last_name, self.first_name, self.second_name


@classmethod
def from_raw_data(cls, data: Dict) -> 'Person':
    jobs = [PersonJob.from_raw_data(job_data) for job_data in data['jobs']]
    return cls(
        id=data['id'],
        first_name=data['first_name'],
        second_name=data['second_name'],
        last_name=data['last_name'],
        jobs=jobs
    )


@classmethod
def validation_schema(cls) -> Any:
    return {
        'type': 'object',
        'properties': {
            'id': {
                'type': 'integer'
            },
            'first_name': {
                'type': 'string',
                'minLength': 1
            },
            'second_name': {
                'type': 'string',
                'minLength': 1
```

```
            },
            'last_name': {
                'type': 'string',
                'minLength': 1
            },
            'jobs': {
                'type': 'array',
                'items': PersonJob.validation_schema()
            }
        },
        'additionalProperties': False,
        'required': ['id', 'first_name', 'second_name', 'last_name', 'jobs']
    }


@dataclass(eq=True, frozen=True)
class Message:
    id: int
    text: str
    creation_time: datetime
    channel_id: str
    messenger: str
```

message_processing/message_processor.py

```
import re
from collections import defaultdict
from typing import List, Dict, Type, Tuple, Set


from jellyfish import jaro_winkler
```

```
from core.classification import BayesianClassifier, DecisionTreeClassifier
from core.classification.classifier import Classifier
from core.common import Message, Person, SINGLE_WORD
from message_processing.person_credential_case import CREDENTIALS_CASES


def filter_people_by_passport_info(people: List[Person], message: Message) -> List[Person]:
    filtered_people: List[Person] = []
    for cred_case in CREDENTIALS_CASES:
        for word_group in re.findall(cred_case.regexp, message.text, re.UNICODE):
            group_single_words: List[str] = re.findall(SINGLE_WORD, word_group, re.UNICODE)
            for person in people:
                if person in filtered_people:
                    continue
                is_presented_mark: bool = True
                respective_data = zip(cred_case.data_getters, cred_case.probabilities,
group_single_words)
                for cred_func, proba, single_word in respective_data:
                    if jaro_winkler(cred_func(person).lower(), single_word.lower()) < proba:
                        is_presented_mark = False
                        break
                if is_presented_mark:
                    filtered_people.append(person)
    return filtered_people


CLASSIFIERS: Tuple[Type[Classifier], ...] = (
    BayesianClassifier,
    # DecisionTreeClassifier,  # TODO
)
```

```
def process_all_people_per_message(people: List[Person],
                                   message: Message,
                                   classifiers_classes: List[Type[Classifier]] = CLASSIFIERS) ->
List[Person]:
    people_by_full_info = filter_people_by_passport_info(people, message)
    classified_people_ids: Set[int] = set()
    for classifier_class in classifiers_classes:
        classifier = classifier_class(people_by_full_info, message)
        classified_local = classifier.process()
        classified_local_ids = {person.id for person in classified_local}
        if not classified_people_ids:
            classified_people_ids = classified_local_ids
        else:
            classified_people_ids.intersection_update(classified_local_ids)
    classified_people: List[Person] = [person for person in people if person.id in
classified_people_ids]
    return classified_people



def get_message_person_mapping(people: List[Person], messages: List[Message]) ->
Dict[Message, List[Person]]:
    people_per_message: Dict[Message, List[Person]] = defaultdict(default_factory=list)
    for message in messages:
        people_per_message[message] = process_all_people_per_message(people, message)
    return people_per_message
```

message_processing/person_credential_case.py

```
from dataclasses import dataclass
from typing import List, Callable
```

```
from core.common import Person



@dataclass
class PersonCredentialsCase:
    regexp: str
    data_getters: List[Callable[[Person], str]]
    probabilities: List[float]


    def __post_init__(self) -> None:
        if len(self.data_getters) != len(self.probabilities):
            raise ValueError('Getters and their respective probabilities collections must have the
same length')



CREDENTIALS_CASES: List[PersonCredentialsCase] = [
PersonCredentialsCase(
        r'\w+\s+\w\.\s*\w\.',
        [
            lambda person: person.last_name,
            lambda person: person.first_name[0],
            lambda person: person.second_name[0]
        ],
        [0.9, 0.99, 0.99]
    ),
    PersonCredentialsCase(
        r'\w\.\s*\w\.\s*\w+',
        [
            lambda person: person.first_name[0],
```

```
            lambda person: person.second_name[0],
            lambda person: person.last_name
        ],
        [0.99, 0.99, 0.9]
    ),
    PersonCredentialsCase(
        r'\w+\s+\w+\s+\w\.',
        [
            lambda person: person.last_name,
            lambda person: person.first_name,
            lambda person: person.second_name[0]
        ],
        [0.9, 0.9, 0.99]
    ),
    PersonCredentialsCase(
        r'\w+\s+\w\.\s*\w+',
        [
            lambda person: person.first_name,
            lambda person: person.second_name[0],
            lambda person: person.last_name
        ],
        [0.9, 0.99, 0.9]
    ),
    PersonCredentialsCase(
        r'\w+\s+\w\s+\w',
        [
            lambda person: person.last_name,
            lambda person: person.first_name[0],
            lambda person: person.second_name[0]
        ],
```

```
            [0.9, 0.99, 0.99]
        ),
        PersonCredentialsCase(
            r'(?=(\w+\s+\w+\s+\w+))\S+\s+',
            [
                lambda person: person.last_name,
                lambda person: person.first_name,
                lambda person: person.second_name
            ],
            [0.9, 0.9, 0.9]
        ),
        PersonCredentialsCase(
            r'(?=(\w+\s+\w+\s+\w+))\S+\s+',
            [
                lambda person: person.first_name,
                lambda person: person.second_name,
                lambda person: person.last_name
            ],
            [0.9, 0.9, 0.9]
        ),
    ]
```

models/substitution.py

```
import json
from typing import List

from sqlalchemy import Column, Integer, Text

from flask_db import db
```

```
class SubstitutionModel(db.Model):
    __tablename__ = 'substitution'

    id = Column(Integer, primary_key=True)
    target = Column(Text, unique=True, nullable=False)
    substitutions_ = Column(Text, nullable=False)

    @property
    def substitutions(self) -> List[str]:
        return json.loads(self.substitutions_)

    @substitutions.setter
    def substitutions(self, value: List[str]) -> None:
        self.substitutions_ = json.dumps(value)

    @classmethod
    def from_raw_substitutions(cls, target: str, substitutions: List[str]) -> 'SubstitutionModel':
        return cls(
            target=target,
            substitutions_=json.dumps(substitutions)
        )
```

Додаток Б

Тексти специфікації прикладного програмного інтерфейсу

Інформаційна система аналізу персоналізованих відгуків учасників навчального процесу

(Найменування програми (документа))

DVD-R

(Вид носія даних)

8 арк, 8 Кб

(Обсяг програми (документа) , арк.,) Кб)

Київ – 2020 року

```
openapi: 3.0.0
servers:
  - description: Campus - Analyzer API contract
    url: https://virtserver.swaggerhub.com/eugene_kells/Diploma_API/1.0.0
info:
  description: API contract for Campus-Analyzer interactions
  version: "1.0.0"
  title: Feedbacks Analyzer contract
  contact:
    email: zheniapivovarenko@ukr.net
tags:
  - name: business
    description: Business-users access
paths:
  /analyze:
    post:
      tags:
        - business
      summary: Performs a search in messenegers channels saccording to provided
parameters
      operationId: searchFeedbacks
      responses:
        '200':
          description: Analysis was successfully completed
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/MessageResponse'
        '422':
          $ref: "#/components/responses/InvalidData"
      requestBody:
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/AnalysisRequest'
        description: Data to be provided to perform analysis
  /substitution:
    get:
      tags:
        - business
      operationId: getTags
      responses:
        '200':
          description: Data was successfully sent
          content:
            application/json:
              schema:
```

```
              type: array
              items:
                $ref: "#/components/schemas/SubstitutionResponse"
          '422':
            $ref: "#/components/responses/InvalidData"
      post:
        tags:
          - business
        operationId: createTag
        responses:
          '201':
            description: Tag successfully created
          '422':
            $ref: "#/components/responses/InvalidData"
        requestBody:
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/TagCreateRequest'
      delete:
        tags:
          - business
        operationId: deleteTag
        parameters:
          - in: query
            name: subst_id
            schema:
              type: integer
            required: true
            description: ID of substitution that is needed to be deleted
        responses:
          '200':
            description: Tag successfully deleted
          '400':
            $ref: "#/components/responses/DataNotFound"
          '422':
            $ref: "#/components/responses/InvalidData"


components:
  schemas:
    AnalysisRequest:
      type: object
      required:
        - people
        - sources
      properties:
        people:
          type: array
          items:
            $ref: "#/components/schemas/PersonInfo"
```

```
    sources:
      type: array
      items:
        anyOf:
          - $ref: '#/components/schemas/TelegramRequest'
PersonInfo:
  type: object
  description: Info about certain person
  required:
    - id
    - first_name
    - second_name
    - last_name
    - jobs
  properties:
    id:
      type: integer
      example: 100500
      description: Unique id to identify person in received set of people
    first_name:
      type: string
      example: John
      description: First name of a person
    second_name:
      type: string
      example: Richard
      description: Middle name of a person
    last_name:
      type: string
      example: Simpson
      description: Last name of a person
    jobs:
      type: array
      items:
        $ref: "#/components/schemas/PersonJob"
PersonJob:
  type: object
  description: Set of jobs that the person worked on
  required:
    - faculty
    - cathedra
    - subject
  properties:
    faculty:
      type: string
      example: Faculty of information techologies
      description: The faculty the person worked on
    cathedra:
      type: string
      example: Cathedra of applied informatics
      description: The cathedra the person worked on
```

```
    subject:
      type: string
      example: Mathematical analysis
      description: The subject the person was teaching
MessageResponse:
  type: object
  required:
    - id
    - text
    - mentioned_people
    - creation_time
    - channel_id
    - messenger
  properties:
    id:
      type: string
      example: 100500
      description: Unique identifier for each message to be identified
    text:
      type: string
      example: In the land far away, there was a kingdom.
      description: Message text itself
    mentioned_people:
      type: array
      items:
        type: integer
        example: 100500
        description: Person id
    creation_time:
      type: string
      format: date-time
      example: "1996-12-19T16:39:57-08:00"
      description: Timestamp when the message was created
    channel_id:
      type: string
      example: kpi_666
      description: The channel id of the channel the message was extracted from
    messenger:
      type: string
      example: telegram
      description: Messenger that the message was extracted from
TelegramRequest:
  type: object
  required:
    - id
    - channels
    - start_time
    - end_time
  properties:
    id:
      type: string
```

```
      enum:
        - telegram
    channels:
      type: array
      items:
        type: string
        example: kpi_777
    start_time:
      type: string
      format: date-time
      example: "1996-12-19T16:39:57-08:00"
      description: Start time point for messages extraction
    end_time:
      type: string
      format: date-time
      example: "1996-12-19T16:39:57-08:00"
      description: End time point for messages extraction
SubstitutionResponse:
  type: object
  properties:
    id:
      type: integer
      example: 100500
      description: Identifier of a tag in DB
    target:
      type: string
      example: Faculty of applied sciences
      description: Tag with association
    substitutions:
      type: array
      items:
        type: string
        example: FAS
TagCreateRequest:
  type: object
  required:
    - substitution_data
  properties:
    substitution_data:
      type: object
      required:
        - target
        - substitutions
      properties:
        target:
          type: string
        substitutions:
          type: array
          items:
            type: string
```

```
    ErrorResponse:
      type: string
      description: Human-readable message to find out the error

  responses:
    InvalidData:
      description: Invalid data was sent, check the format and types of the data in payload
      content:
        text/html:
          schema:
            $ref: "#/components/schemas/ErrorResponse"
    DataNotFound:
      description: Data was not found according to parameters provided
      content:
        text/html:
          schema:
            $ref: "#/components/schemas/ErrorRespon
```

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО”
Кафедра автоматизованих систем обробки інформації та управління

УЗГОДЖЕНО	**ЗАТВЕРДЖУЮ**
Керівник проєкту	**В.о. завідувача кафедри**
_____ *Олексій ФІНОГЕНОВ*	_____ *Олександр ПАВЛОВ*
(підпис) (вл. ім’я, прізвище)	(підпис) (вл. ім’я, прізвище)
“13” квітня 2020 р.	“14” квітня 2020 р.

Інформацйна система аналізу персоналізованих відгуків учасників навчального процесу

ТЕХНІЧНЕ ЗАВДАННЯ

Шифр *ДП 6101.01.000 ТЗ*

на 11 сторінках

Київ – 2020 року

ЗМІСТ

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Повне найменування системи та її умовне позначення

Повне найменування системи – інформаційна система аналізу персоналізованих відгуків учасників навчального процесу.

Умовне позначення системи – "ISEPF" (скорочено від англ. "Information System for Analysis of Educational Process Participants' Personalised Feedbacks").

1.2 Найменування організації-замовника та організацій-учасників робіт

Замовником розробки системи є кафедра автоматизованих систем обробки інформації та управління факультету інформатики та обчислювальної техніки Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського". Адреса замовника: Україна, м. Київ, просп. Перемоги, 37, корп. 18.

Розробником системи є студент групи ІС-61 кафедри автоматизованих систем обробки інформації та управління факультету інформатики та обчислювальної техніки Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського" Півоваренко Євгеній Павлович.

1.3 Перелік документів, на основі яких створюється система

Розробка даної системи проводиться на основі наказу №1081-с від 07.05.2020 року факультету інформатики та обчислювальної техніки Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського".

1.4 Планові терміни початку і закінчення робіт зі створення системи

Плановий строк початку робіт зі створення системи – 15 березня 2020 року.

Плановий строк закінчення робіт зі створення системи – 30 травня 2020 року.

2 ПРИЗНАЧЕННЯ І ЦІЛІ СТВОРЕННЯ СИСТЕМИ

2.1 Призначення системи

Призначенням системи є автоматизація процесу визначення отримувача анонімного персоналізованого неструктурованого відгуку, використовуючи масив даних про потенційних отримувачів.

2.2 Цілі створення системи

Для розробки системи необхідно поставити наступні цілі:

- створення та запровадження у системі алгоритму аналізу персоналізованих неструктурованих анонімних відгук;
- створення прикладного програмного інтерфейсу, що дозволить проводити інтеграцію з автоматизованими системами формування та надсилання масиву даних про осіб для аналізу.

Для вирішення поставлених цілей необхідно вирішити наступні задачі:

- розробити алгоритм для визначення цільових осіб у неструктурованих документах;
- створити тестовий навчальний набір даних відгуків;
- провести тестування ефективності розробленого алгоритму за допомогою сформованого набору даних;
- визначити контракт прикладного програмного інтерфейсу для взаємодії системи з системами-клієнтами (автоматичного формування та надсилання даних про осіб для аналізу);
- реалізувати систему для автоматичної агрегації та аналізу відгуків; забезпечити можливість агрегації відгуків з інформаційно-комунікаційної системи Telegram;
- провести тестування розробленої системи на відповідність поставленим вимогам.

3 ХАРАКТЕРИСТИКА ОБ'ЄКТА АВТОМАТИЗАЦІЇ

З розвитком інформаційних технологій підвищилась і можливість взаємодії між учасниками навчального процесу, в тому числі й брати участь в спільнотах, що дозволить ділитися оцінками про інших учасників навчального процесу, що є важливим фактором для покращення якості освіти. Також причиною цього стала недостатня залученість в процес оцінки якості навчання вже існуючих навчальних систем, таких Moodle. З цих причин, великої популярності набули групи для анонімних відгуків про навчання або ж окремих учасників навчального процесу, що дозволяють ділитися оцінкою одразу для широкої групи осіб, не втрачаючи при цьому можливості залишитись анонімним, що є важливим психологічним аспектом за такого роду взаємодії.

Виходячи з вищеописаного, постає завдання визначення осіб, що є отримувачами конкретного відгуку, згаданого вище. Прийнявши до уваги те, що такі відгуки мають неструктуровану природу, а також те, що для визначення осіб необхідно використовувати великі об'єми даних, такий процес є досить затратнім і потребує зусиль пропорційно до кількості наданих відгуків. Саме тому постає питання автоматизації процесу визначення отримувачів відгуку, що дозволить економити ресурси, а згодом, з потенційним покращенням таких систем, і застосувати даний підхід до інших областей класифікації текстів та визначення іменованих сутностей у текстах.

Отже, маємо об'єктом автоматизації визначення отримувачів неструктурованих анонімних персоналізованих відгуків.

4 ВИМОГИ ДО ПРОГРАМНОГ ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

До системи ставляться наступні функціональні вимоги:

- система повинна здійснювати агрегацію та аналіз відгуків з інформаційно-комунікаційної мережі Telegram на основі даних, отриманих для аналізу від Користувача (система-клієнта);
- система НЕ повинна зберігати дані, отримані від Користувача, після завершення аналізу;
- у випадку встановлення кількох осіб з однаковими іменами, іменами по-батькові та прізвищами особами, згаданими у повідомленні, система повинна НЕ включати жодного з них у результати аналізу даного повідомлення;
- у випадку недостатності або некоректності формату даних, визначених у контракті прикладного програмного інтерфейсу та надісланих Користувачем, система повинна повідомити Користувача про це відповідним повідомленням у відповідь.

4.2 Вимоги до надійності

Вимоги до надійності системи не висуваються.

4.3 Умови експлуатації

Вимоги до умов експлуатації системи не висуваються.

4.4 Вимоги до складу і параметрів технічних засобів

В склад технічних засобів для розгортання системи повинні входити наступні компоненти:

- 32-розрядний (x86) або 64-розрядний (x64) процесор із тактовою частотою 1 ГГц або та більше;
- 1 гігабайт (ГБ) RAM (для 32-розрядної версії) або 2 ГБ (для 64-розрядної версії);
- 1 ГБ вільного місця на жорсткому диску.

Дана система може бути розгорнута в наступних операційних системах:

- Linux;

- Unix.

Для розгортання системи та її коректного функціонування, в операційній системі повинна бути встановлена мова програмування Python не нижче версії 3.7.7 і повинен бути можливим доступ до мережі для зв'язку з Користувачем.

5 СТАДІЇ І ЕТАПИ РОЗРОБКИ

Виділимо етапи проведення розробки системи:

Таблиця 5.1 – Стадії розробки системи

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	Визначення цілей та задач розробки системи	25.03.2020	
2.	Розробка сценарію роботи системи	01.04.2020	
3.	Узгодження методів розв'язання поставлених задач	20.04.2020	
4.	Узгодження інтерфейсу користувача	20.04.2020	
5.	Розробка програмної реалізації системи	01.05.2020	
6.	Тестування системи	13.05.2020	
7.	Виправлення недоліків та покращення роботи системи	22.05.2020	
8.	Кінцева здача розробленої системи	30.05.2020	

6 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

6.1 Види випробувань

Для забезпечення якості реалізації системи, необхідно провести наступні випробування:

- Випробування згідно функціональних вимог, поставлених до системи.

Виділимо наступні етапи для випробування функціональних вимог та очікуваний результат для кожного випробування відповідно:

Таблиця 6.1 – Стадії розробки системи

№ з/п функціональної вимоги	Очікуваний результат
1	Система повинна провести аналіз даних по усіх джерелах даних, вказаних Користувачем у запиті на проведення аналізу, та повернути агрегований результат по усіх вказаних джерелах.
2	Система не повинна записати дані, отримані від Користувача для проведення аналізу, в жодне постійне сховище даних (база даних, файл тощо).
3	При встановленні однакової ймовірності згадування осіб у документі, система повинна не включити жодного з них як згадуваної особи.

Продовження таблиці 6.1

№ з/п функціональної вимоги	Очікуваний результат
4	Для будь-якого запиту від Користувача до системи за наявності неправильного формату або змісту даних, надісланих Користувачем, система повинна повідомити про це відповідним повідомленням. У разі правильності даних, система повинна провести необхідну операцію та повідомити про її успішне виконання.

UNICHECK

Власник документу:
Попенко Володимир Дмитрович

ID перевірки:
1003947937

Дата перевірки:
11.06.2020 01:37:44 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
12.06.2020 01:23:21 EEST

ID користувача:
77149

Назва документу: Pivovarenko_is61

ID файлу: 1003963098 Кількість сторінок: 105 Кількість слів: 15237 Кількість символів: 120689 Розмір файлу: 1.45 MB

5.6% Схожість

Найбільша схожість: 2.24% з джерело бібліотеки. ID файлу: 1003935263

3.59% Схожість з Інтернет джерелами	74	Page 107
4.73% Текстові збіги по Бібліотеці акаунту	281	Page 108

0.66% Цитат

Цитати	2	Page 109

Вилучення переліку посилань вимкнено

0% Вилучень

Вилучений текст відсутній

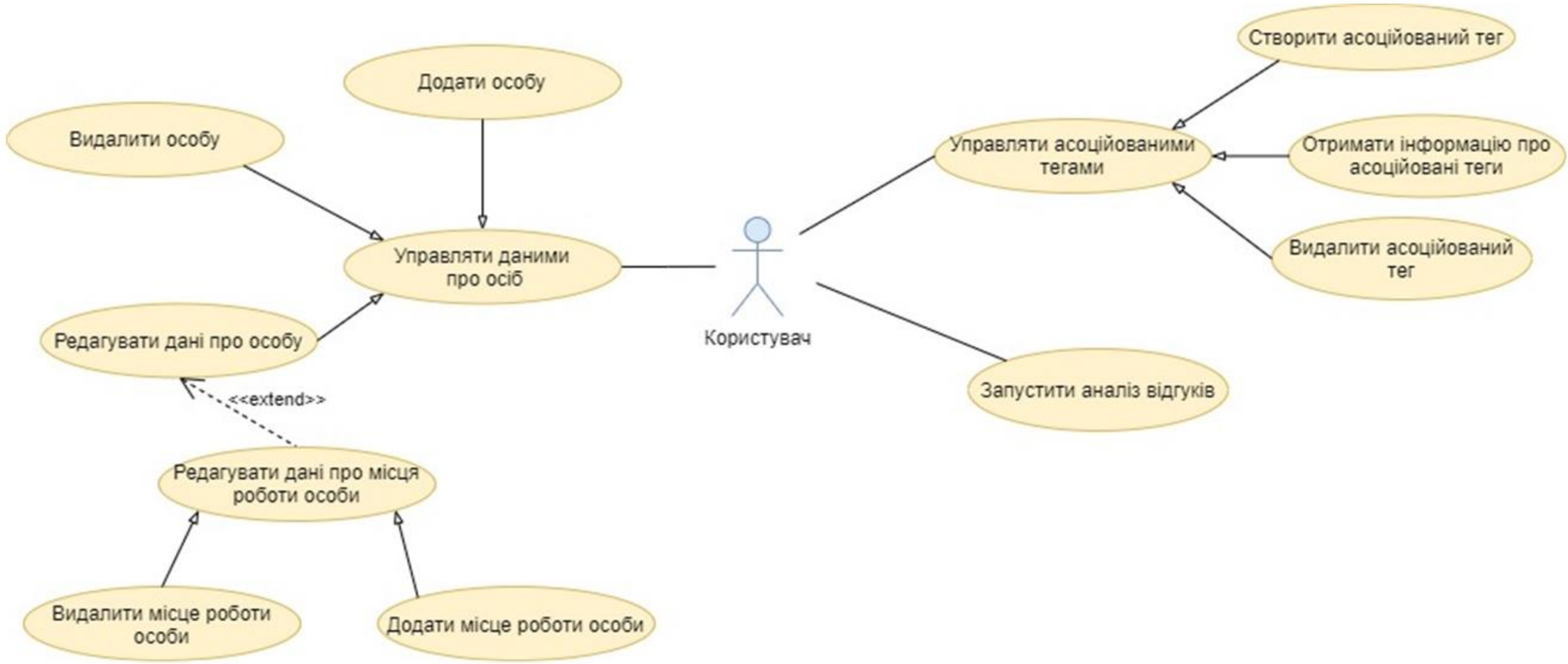
Підміна символів

Заміна символів	6

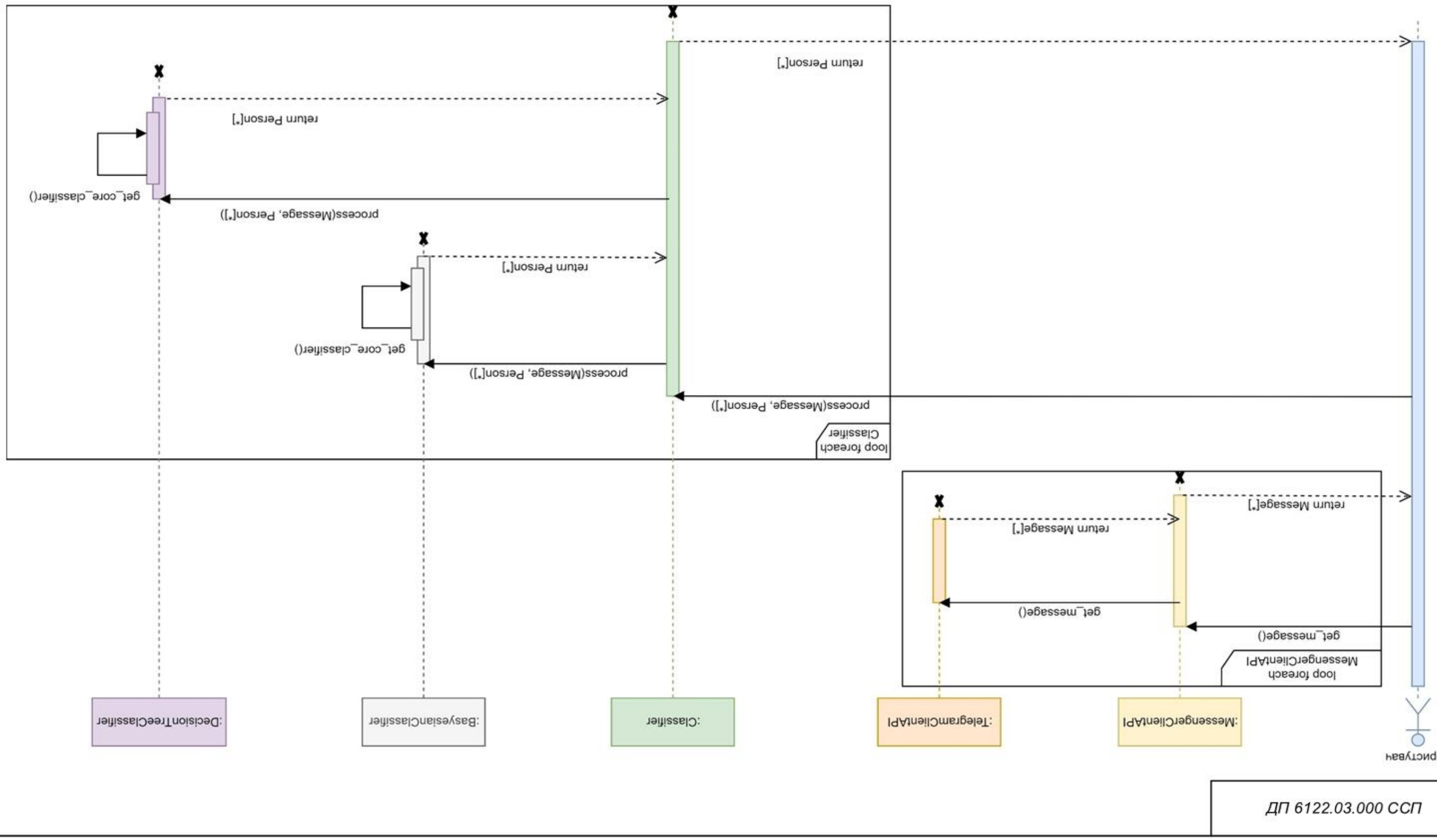
Графічний матеріал
до дипломного проєкту

на тему: Інформаційна система аналізу персоналізованих відгуків учасників навчального процесу

Київ – 2020 року



					ДП 6122.02.000 ССВ			
					Схема структурна варіантів використання	Літера	Маса	Масштаб
Зм.	Арк.	№ документа	Підпис	Дата				
Розробив		Півоваренко Є.П.				Аркуш 1	Аркушів 1	
Перевірив		Фіногенов О.Д.						
Т. кон.								
					Інформаційна система аналізу персоналізованих відгуків учасників навчального процесу	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61		
Н. кон.		Телишева Т.О.						
Затвердив		Фіногенов О.Д.						



Зм.	Арк	№ докумен та	Підпис	Дата					
Розробив		Пономаренко Є.Л.							
Перевірив		Фіногенов О.Д.							
Т. контр.									
Н. контр.		Тепцевва Т.О.							
Затвердив		Фіногенов О.Д.							

Зм.	Арк	№ докумен та	Підпис	Дата					
Розробив		Пономаренко Є.Л.							
Перевірив		Фіногенов О.Д.							
Т. контр.									
Н. контр.		Тепцевва Т.О.							
Затвердив		Фіногенов О.Д.							

ДП 6122.03.000 ССП

Літера	Маса	Масштаб

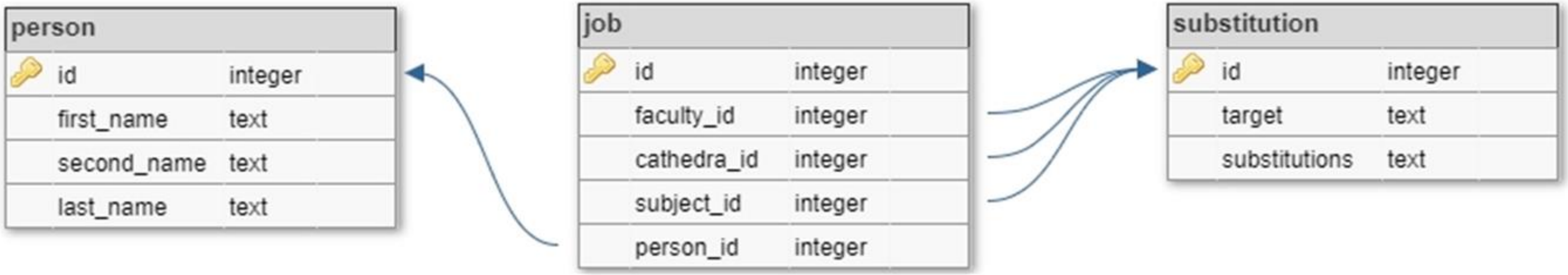
Схема структура послідовності програмного забезпечення

Арк	Аркушів
1	1

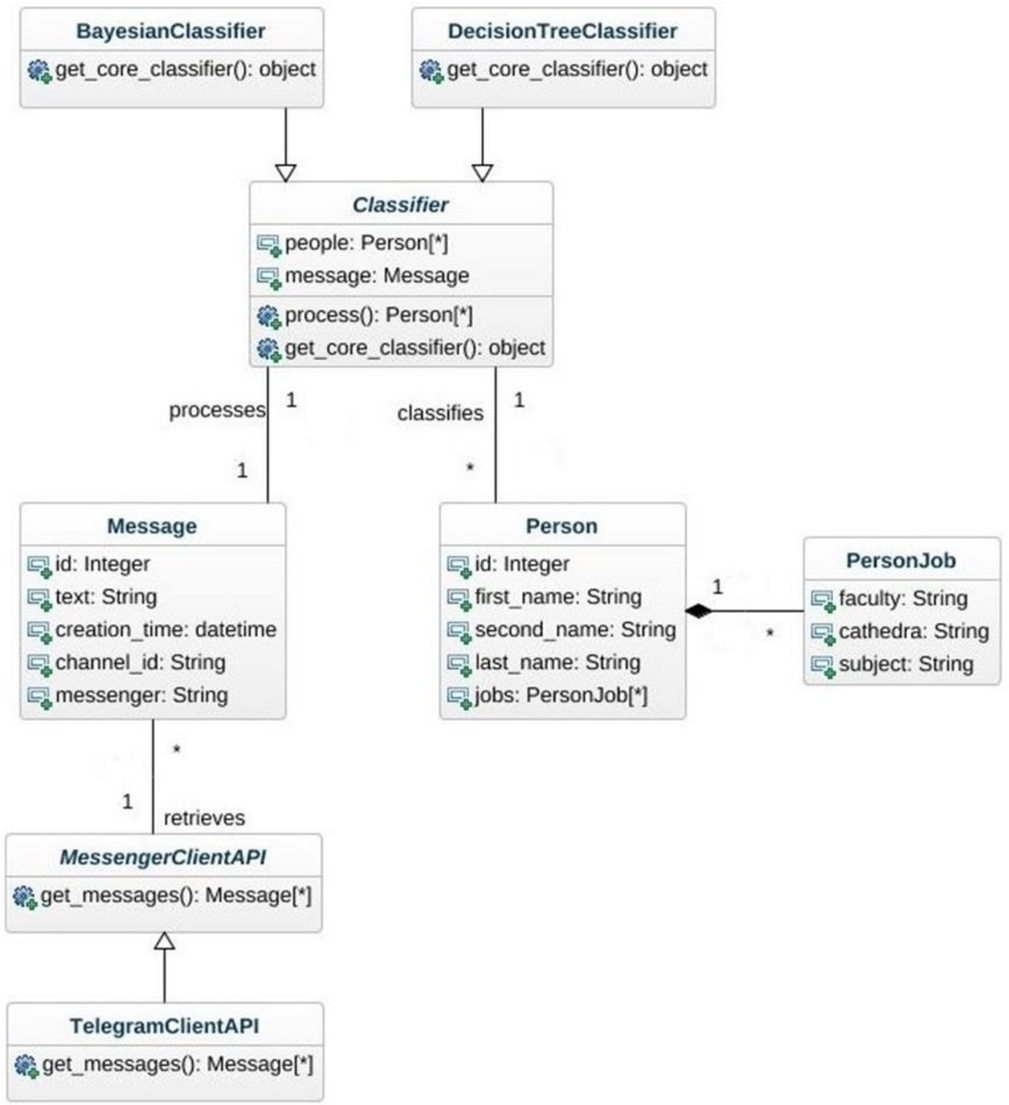
Інформаційна система аналізу персоналізованих відгуків учасників навчального процесу

КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61

ДП 6122.04.000 СБД

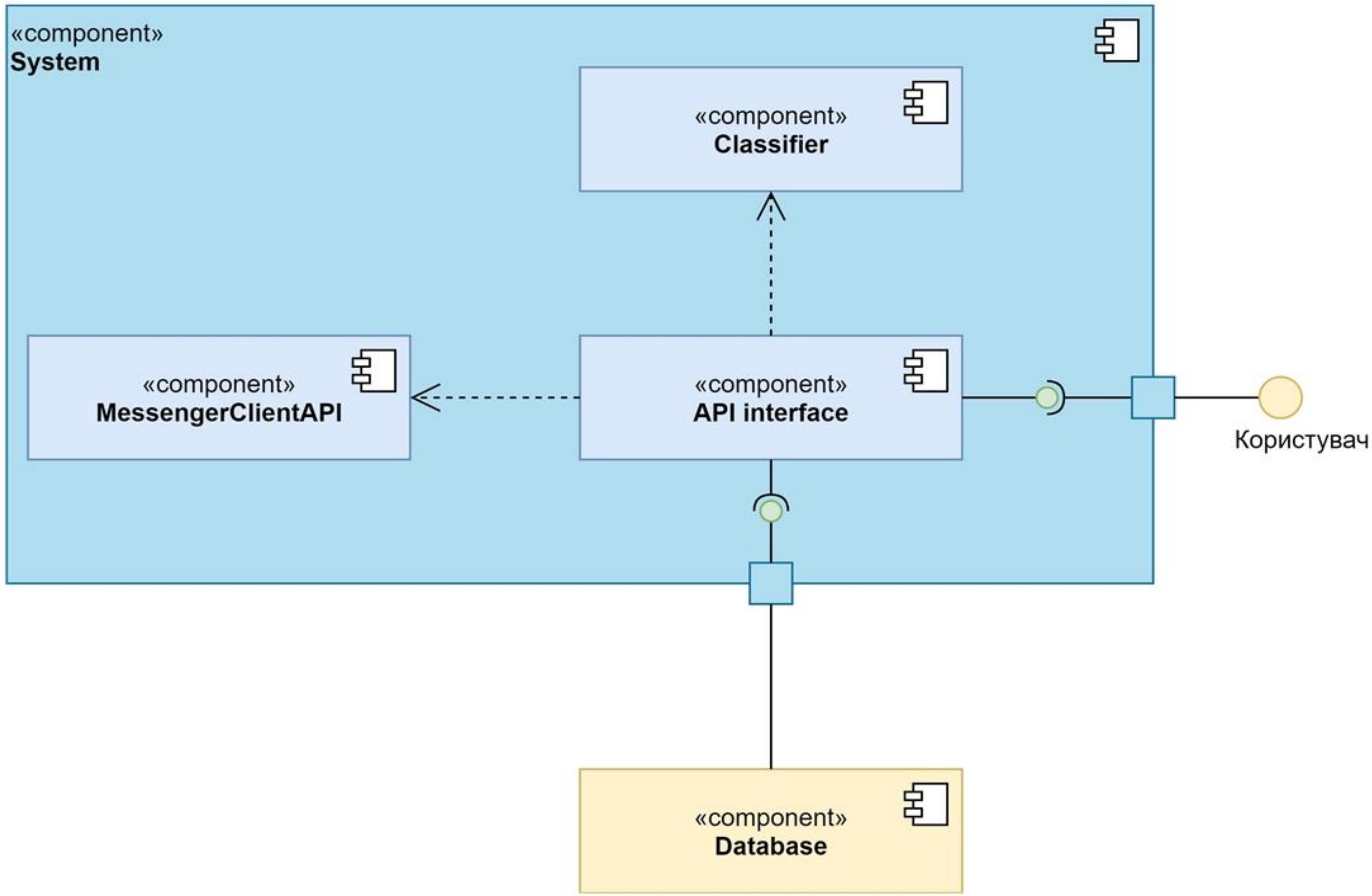


					ДП 6122.04.000 СБД				
Зм.	Арк.	№ документа	Підпис	Дата	Схема бази даних	Літера		Маса	Масштаб
Розробив		Півоваренко Є.П.							
Перевірив		Фіногенов О.Д.				Аркуш 1		Аркушів 1	
Т. кон.					Інформаційна система аналізу персоналізованих відгуків учасників навчального процесу	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61			
Н. кон.		Телишева Т.О.							
Затвердив		Фіногенов О.Д.							



					ДП 6122.05.000 ССК	
					Схема структурна класів програмного забезпечення	Лит. / Маса / Масштаб
Зм.	Арк.	№ докум.	Підп.	Дата		
Розроб.		Півоваренко Є.П.				Аркуш 1 / Аркушів 1
Перев.		Фіногенов О.Д.				
Т. Кон.					Інформаційна система аналізу персоналізованих відгуків учасників навчального процесу	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61
Н. Кон.		Тєлишева Т.О.				
Затв.		Фіногенов О.Д.				

ДП 6122.06.000 ССК



					ДП 6122.06.000 ССК			
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна компонентів програмного забезпечення	Літера	Маса	Масштаб
Розробив		Півоваренко Є.П.						
Перевірив		Фіногенов О.Д.						
Т. кон.						Аркуш 1		Аркушів 1
					Інформаційна система аналізу персоналізованих відгуків учасників навчального процесу	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61		
Н. кон.		Телишева Т.О.						
Затвердив		Фіногенов О.Д.						

POST localhost:6666/substitution

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
    "substitution_data": {
        "target": "Математичний аналіз",
        "substitutions": null
    }
}
```

Body Cookies Headers (4) Test Results Status: 422 UNPROCESSABLE ENTITY

Pretty Raw Preview Visualize JSON

```
"Input values are incorrect"
```

DELETE localhost:6666/substitution?substId=3

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
substId	3	
Key	Value	Description

Body Cookies Headers (4) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```
"Substitution was successfully deleted"
```

					ДП 6122.07.000 КЗ		
Зм.	Арк.	№ документа	Підпис	Дата	Креслення вигляду звітних форм	Літера	Маса / Масштаб
Розробив		Півоваренко Є.П.					
Перевірив		Фіногенов О.Д.				Аркуш 1	Аркушів 3
Т. кон.							
Н. кон.		Телишева Т.О.			Інформаційна система аналізу персоналізованих відгуків учасників навчального процесу	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61	
Затвердив		Фіногенов О.Д.					

GET localhost:6666/substitution

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (4) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON


```
[
    {
        "id": 13,
        "target": "Факультет електроніки",
        "substitutions": [
            "ФЭЛ"
        ]
    },
    {
        "id": 19,
        "target": "Математичний аналіз",
        "substitutions": [
            "Матан",
            "Вишка",
            "Вышмат",
            "МА"
        ]
    }
]
```


POST localhost:6666/analysis Send Save

Body Cookies Headers (4) Test Results Status: 200 OK Time: 920 ms Size: 3.93 KB Save Response

Pretty Raw Preview Visualize JSON


```
[
    {
        "id": 474,
        "text": "#ФБТ 5 курс \n\nБендюг В.І. Викладає сталий розвиток. На час карантину запропонував декілька варіантів отримання балів, що включали в себе як різноманітні курси на вибір, так і виконання інших завдань (реферат/доповіді тощо), тому кожен міг обрати те, що йому до душі. Останні новини дізнаємося як з особистого сайту викладача, так і в чаті предмету в телеграм-каналі. Онлайн-лекції також наявні, викладач подає цікаву та актуальну інформацію",
        "creation_time": "2020-04-16T19:22:14+00:00",
        "channel_id": "kpi_777",
        "messenger": "telegram",
        "mentioned_people": [
            1
        ]
    },
    {
        "id": 473,
        "text": "#ФБТ 5 курс \nСамойленко О.В. Топ викладач з інтелектуальної власності. Лекції цікаві, на дистанційному навчанні проводить лекції в зум, запропонував допомогу у, питанні створення власного патенту, також скинув список курсів із предмету. 👍",
        "creation_time": "2020-04-16T19:21:20+00:00",
        "channel_id": "kpi_777",
        "messenger": "telegram",
        "mentioned_people": []
    }
]
```


					ДП 6122.07.000 КЗ	
Зм.	Арк.	№ документа	Підпис	Дата	Креслення вигляду звітних форм	Літера: — / Маса: — / Масштаб: —
Розробив		Півоваренко Є.П.				Аркуш 2 / Аркушів 3
Перевірив		Фіногенов О.Д.			Інформаційна система аналізу персоналізованих відгуків учасників навчального процесу	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61
Т. кон.						
Н. кон.		Тєлишева Т.О.				
Затвердив		Фіногенов О.Д.				

POST localhost:6666/substitution

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
    "substitution_data": {
        "target": "Математичний аналіз",
        "substitutions": [
            "Матан",
            "Вишка",
            "Вышмат",
            "МА"
        ]
    }
}
```

Body Cookies Headers (4) Test Results Status: 201 CREATED

Pretty Raw Preview Visualize JSON

```
"Substitution successfully set"
```

DELETE localhost:6666/substitution?substId=3

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
substId	3	
Key	Value	Description

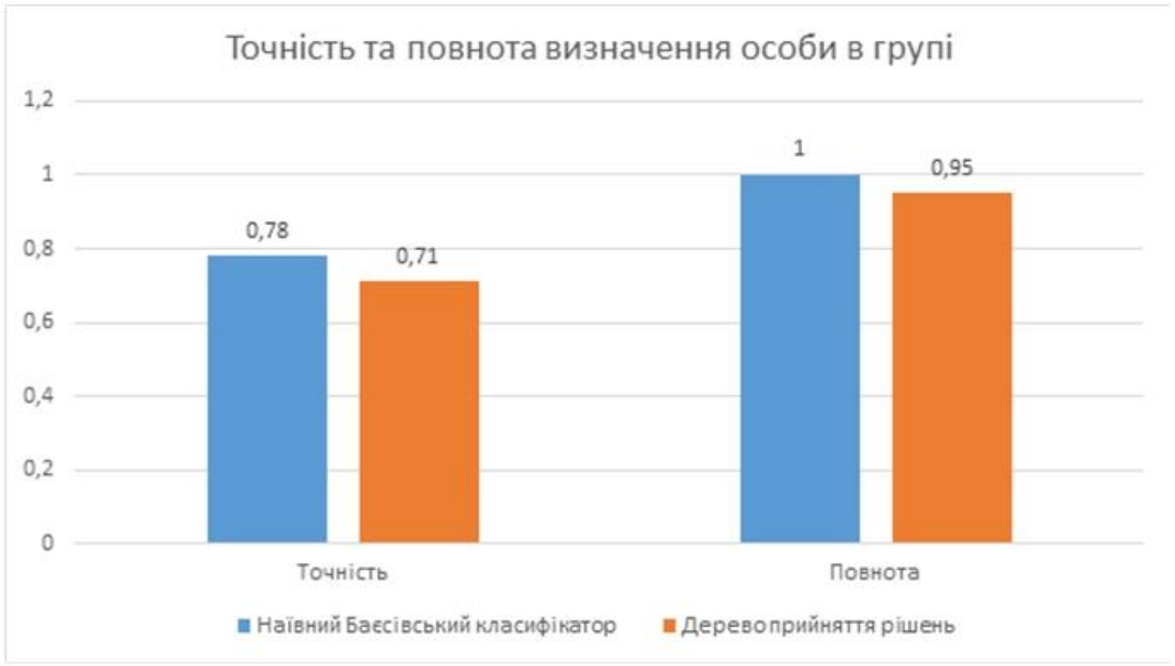
Body Cookies Headers (4) Test Results Status: 400 BAD REQUEST

Pretty Raw Preview Visualize JSON

```
"Substitution with id 3 does not exist"
```

Зм.	Арк.	№ документа	Підпис	Дата	ДП 6122.07.000 КЗ		
Розробив		Півоваренко Є.П.			Креслення вигляду звітних форм	Літера	Маса / Масштаб
Перевірив		Фіногенов О.Д.				Аркуш 3	Аркушів 3
Т. кон.							
Н. кон.		Телишева Т.О.			Інформаційна система аналізу персоналізованих відгуків учасників навчального процесу	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-61	
Затвердив		Фіногенов О.Д.					

Рішення з математичного забезпечення



Демонстраційний плакат до дипломного проекту

„Інформаційна система аналізу персоналізованих відгуків учасників навчального процесу ”

Виконав студент гр. ІС-61 Півоваренко Є.П.

Керівник ДП Фіногенов О.Д.